# BEHAVIOR OF INFINITELY WIDE NEURAL NETWORKS

HUNG T. C. LE

Abstract. In this expository paper, I explore the basics of the neural tangent kernel, and how it is related to the behavior of fully-connected feedforward neural networks with infinitely wide hidden layers.

Contents



*Date*: December 12, 2025.

## 1. Introduction

Generalization and convergence for neural networks are mysterious. For example, it is unclear whether optimization techniques such as gradient descent can actually facilitate convergence to a global minimum on a highly non-convex empirical loss landscape, and even if it does, it is unclear whether such a minimizer can generalize well beyond the training set. However, as it turns out, these questions can be answered in the special regime of infinitely wide fully-connected feedforward neural networks. It takes a change of point of view: to view gradient descent in the function space, instead of in parameter space, and magical convergences in the wide limit that reduce the optimization process to a kind of kernel regression and greatly simplify analysis. In this expository paper, I will explore the basics of that kernel — the neural tangent kernel, and how it is used to analyze the behavior of fully-connected feedforward neural networks with infinitely wide hidden layers. Some topics that are introduced along the way are Gaussian processes, reproducing kernel Hilbert spaces and kernel regression. Most, if not all of the material here can be found in the original Neural Tangent Kernel paper [1].

**Remark 1.1.** While writing this paper, I viewed it as an educational vehicle, mostly for myself and hopefully for the reader. Apologies are in order for possibly verbose explanations — they helped me with my intuition.

## 2. Infinitely wide neural networks at initialization

### 2.1. Gaussian processes.

Let's start with some basics.

**Definition 2.1** (Normal Random Variable)**.** A random variable $X$ taking values in $\mathbb{R}$ is **normal** with mean $\mu \in \mathbb{R}$ and variance $\Sigma \in \mathbb{R}_+$ if its probability density function is

$$f(x) = \frac{1}{\sqrt{2\pi\Sigma}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\Sigma}\right).$$

The degenerate case is when $\Sigma = 0$, i.e. $X$ is constant (a.s.). We denote $X \sim \mathcal{N}(\mu, \Sigma)$.

Let's bring up the number of dimensions.

**Definition 2.2** (Multivariate Normal Random Variable)**.** The collection $X = (X^{(1)}, \ldots, X^{(d)})$ of $d$ real-valued random variables is **multivariate normal** with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ if there exists $A \in \mathbb{R}^{d\times s}$ such that

$$X = AZ + \mu,$$

where $Z = (Z_1, \ldots, Z_s)$ has $Z_i \overset{i.i.d.}{\sim} \mathcal{N}(0,1)\ \forall\, i \in [s]$, and that $\Sigma = AA^\intercal$.

One should think of $X$ as a random variable in $\mathbb{R}^d$ of interacting coordinates via transformation $A$. Note that $\Sigma$ is automatically positive semidefinite. When it is positive definite, it is *non-degenerate*, exactly when $\ker(A^\intercal) = \{0\}$, i.e. $A$ is surjective. In that case, the probability densition function of $X$ exists and is

$$f(x) = \frac{1}{(2\pi \det\Sigma)^{d/2}} \exp\left(-\frac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)\right),$$

reminiscent of the 1-dimensional case. We similarly denote $X \sim \mathcal{N}(\mu, \Sigma)$.

Terminology wise, "Gaussian" is used interchangeably with "normal." We can further extend this to an uncountably infinite setting, where the same intuition should follow. Let's further bring up the number of dimensions.

**Definition 2.3** (Gaussian Process)**.** The collection $Z = (Z(x) \in \mathbb{R} : x \in \mathcal{X})$ of real-valued random variables indexed by $\mathcal{X}$ is a **Gaussian process** of mean function $\mu : \mathcal{X} \to \mathbb{R}$ and covariance function $\Sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ if every finite collection of those random variables is a multivariate normal of corresponding mean and covariance, i.e. for any $x^{(1)}, \ldots, x^{(m)}$,

$$(Z(x^{(1)}), \ldots, Z(x^{(m)})) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu} \in \mathbb{R}^d, \boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ with $\boldsymbol{\mu}_i = \mu(x^{(i)}), \boldsymbol{\Sigma}_{ij} = \Sigma(x^{(i)}, x^{(j)}) \; \forall \, i, j \in [m]$. In what follows, we'll abuse notation and write $\mu(x^{(1)}, \ldots, x^{(m)}) = \boldsymbol{\mu}$ and $\Sigma(x^{(1)}, \ldots, x^{(m)}) = \boldsymbol{\Sigma}$ when the context is clear.

We denote $Z \sim \mathcal{GP}(\mu, \Sigma)$. One should now think of $Z$ as a random function $\mathcal{X} \to \mathbb{R}$ following a particular dependence structure.

**Remark 2.4.** We will make statements about convergence (in distribution) of collections of random variables to particular Gaussian processes. We will brush aside questions of existence and uniqueness (and have done so) of $\mathcal{GP}$, especially when the covariance functions $\Sigma$ we talk about are sufficiently nice; and it will suffice to demonstrate convergence (in distribution) of $(Z^{(n)}(x)) \to (Z(x)) \sim \mathcal{GP}$ on a finite set of indices, i.e., for any $x^{(1)}, \ldots, x^{(m)} \in \mathcal{X}$,

$$\begin{aligned}(Z^{(n)}(x^{(1)}), \ldots, Z^{(n)}(x^{(m)})) &\xrightarrow{n \to \infty, d} (Z(x^{(1)}), \ldots, Z(x^{(m)})) \\ &= \mathcal{N}(\mu(x^{(1)}, \ldots, x^{(m)}), \Sigma(x^{(1)}, \ldots, x^{(m)})).\end{aligned}$$

To do this, a tool we will appeal to is

**Theorem 2.5** (Multivariate Central Limit Theorem (Multivariate CLT))**.** *If $X_1$, $\ldots$, $X_n$ are i.i.d. $\mathbb{R}^d$-valued random variables with $\mathbb{E}(X) = \boldsymbol{\mu}$ and $\mathrm{Cov}(X) = \boldsymbol{\Sigma}$, then*

$$\sqrt{n}(\overline{X}_n - \boldsymbol{\mu}) \xrightarrow{n \to \infty, d} N(0, \boldsymbol{\Sigma}),$$

*where*

$$\overline{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$$

*is the sample mean.*

2.2. **Fully-connected feedforward neural networks.** We now describe the fully-connected feedforward neural network (henceforth "neural network") $f : \mathbb{R}^{n_0 = d_{in}} \to \mathbb{R}^{n_L = d_{out}}$ of depth $L$, consisting of $(L-1)$ hidden layers of sizes $n_1, \ldots, n_{L-1}$ and an output layer of size $n_L = d_{out}$, with a sufficiently nice[1] non-linearity $\sigma$. Its output

[1]Lipschitz. Sometimes twice differentiable with bounded second derivative is required.

$f(x) = \tilde{\alpha}^L(x)$ is defined recursively via: for all $0 \leq l \leq L-1$,

$$\alpha^0(x) = x,$$
$$\forall\, i \in [n_{l+1}], \quad \tilde{\alpha}_i^{l+1}(x) = \beta b_i^l + \frac{1}{\sqrt{n_l}} \sum_{j=1}^{n_l} W_{ij}^l \alpha_j^l(x)$$
$$\alpha^{l+1}(x) = \sigma(\tilde{\alpha}^{l+1}(x)).$$

where $\tilde{\alpha}^{l+1}$, being the affine transformation of signals coming from layer $l$, is the pre-activation at layer $l+1$, $\alpha^{l+1}$ is the post-activation at the same layer, and $W$ and $b$ are weight and bias parameters to be initialized and optimized over.

Note two variables that are irrelevant here: $\tilde{\alpha}^0$ isn't present, because we use the raw input as the signal emitted from the first layer, and crucially $\alpha^L$ — we do not have activation on the final layer, and simply use $f(x) = \tilde{\alpha}^L(x)$.

For each configuration of $(n_1, \ldots, n_{L-1})$, initializing all weight and bias parameters by drawing them i.i.d. from $\mathcal{N}(0,1)$ induces a distribution over functions $\left\{\tilde{\alpha}^L : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}\right\}$. We are first interested in the limiting behavior of this distribution as the hidden layer widths are taken to $\infty$. As it turns out, the limiting distribution is a composition of Gaussian processes.

### 2.3. **Convergence of general-depth neural networks to Gaussian processes.**

**Theorem 2.6** ([1], Proposition 1)**.** *In the limit $n_1, \ldots, n_{L-1} \to \infty$ sequentially, the output functions converge in distribution to*

$$f_1, \ldots, f_{d_{out}} \overset{i.i.d.}{\sim} \mathcal{GP}(0, \Sigma^L)$$

*where $\Sigma$ is defined recursively:*

$$\Sigma^1(x, x') = \frac{1}{n_0} x^\intercal x' + \beta^2,$$
$$\Sigma^{l+1}(x, x') = \mathbb{E}_{f \sim \mathcal{GP}(0, \Sigma^l)}[\sigma(f(x))\sigma(f(x'))] + \beta^2.$$

*Proof.* We will prove this result by induction.

**Base case:** $L = 1$. This just means that the computation graph is that for each $\iota \in [d_{out}]$,

$$f_\iota(x) = \tilde{\alpha}_\iota^1(x) = \beta b_\iota^0 + \frac{1}{\sqrt{n_0}} \sum_{\kappa=1}^{n_0} W_{\iota\kappa}^0 x_\kappa.$$

There are two things we need to show: that $f_\iota \sim \mathcal{GP}(0, \Sigma^1)$ and they are independent for $\iota \neq \iota'$.

First, the distributional statement: in this base case, it is possible to make statements without taking limits, because for each $\iota$, and for any $(x^{(1)}, \ldots, x^{(m)})$, we get that

$$\begin{bmatrix} f_\iota(x^{(1)}) \\ f_\iota(x^{(2)}) \\ \vdots \\ f_\iota(x^{(m)}) \end{bmatrix} = \beta b_\iota^0 \mathbf{1} + \frac{1}{\sqrt{n_0}} \sum_{\kappa=1}^{n_0} W_{\iota\kappa}^0 \begin{bmatrix} x_\kappa^{(1)} \\ x_\kappa^{(2)} \\ \vdots \\ x_\kappa^{(m)} \end{bmatrix}.$$

Because $x^{(1)}, \ldots, x^{(m)}$ are fixed, and $W^0_{\iota\kappa} \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$, each summand is an independent multivariate normal. Thus the same is also multivariate normal with mean clearly being 0 and covariance matrix:

$$\begin{aligned}\mathbf{\Sigma} &= \beta^2 \mathbf{1}\mathbf{1}^\intercal + \frac{1}{n_0}\sum_{\kappa=1}^{n_0} \mathrm{Cov}\left(\begin{bmatrix} W^0_{\iota\kappa} x^{(1)}_\kappa \\ W^0_{\iota\kappa} x^{(2)}_\kappa \\ \vdots \\ W^0_{\iota\kappa} x^{(m)}_\kappa \end{bmatrix}\right) \\ &= \beta^2 \mathbf{1}\mathbf{1}^\intercal + \frac{1}{n_0}\sum_{\kappa=1}^{n_0} \begin{bmatrix} x^{(1)}_\kappa \\ x^{(2)}_\kappa \\ \vdots \\ x^{(m)}_\kappa \end{bmatrix} \begin{bmatrix} x^{(1)}_\kappa & x^{(2)}_\kappa & \ldots & x^{(m)}_\kappa \end{bmatrix}\end{aligned}$$

which has entries $\mathbf{\Sigma}_{ij} = \beta^2 + \frac{1}{n_0}\sum_{\kappa=1}^{n_0} x^{(i)}_\kappa x^{(j)}_\kappa = \beta^2 + \frac{1}{n_0} x^{(i)\intercal} x^{(j)} = \Sigma^1(x^{(i)}, x^{(j)})$. Thus we've shown that $f_\iota \sim \mathcal{GP}(0, \Sigma^1)$.

Second, the independence statement: for $\iota \neq \iota'$, they depend on disjoint, independent sets of weights $\{b^0_\iota, W^0_{\iota\cdot}\}$ and $\{b^0_{\iota'}, W^0_{\iota'\cdot}\}$, so they are independent.

We have therefore demonstrated the base case. Note that the statement does not depend on the size of $d_{out}$, and is therefore also valid in the limit case when this output layer width is taken to $\infty$, when it becomes a hidden layer.

**Inductive step:** Consider a network of depth $L+1$, i.e. of $L$ hidden layers and 1 output layer, constructed based off of a network of depth $L$ of $L-1$ hidden layers with 1 output layer, by mapping input $x$ through the subnetwork to get $\tilde{\alpha}^L(x)$, and then through the non-linearity $\sigma$ to give $\alpha^L(x)$, and then an affine transformation to give $\tilde{\alpha}^{L+1}(x)$.

The induction hypothesis gives that as sequentially $n_1, \ldots, n_{L-1} \to \infty$, the preactivations $\tilde{\alpha}^L_\iota$ for $\iota \in [n_L]$ converge in distribution to i.i.d. $\mathcal{GP}(0, \Sigma^L)$. Now we send $n_L \to \infty$. Again, we need to show that for any $\iota \in [n_{L+1} = d_{out}]$, $f_\iota \sim \mathcal{GP}(0, \Sigma^{L+1})$, and they are independent for $\iota \neq \iota'$.

To show the distributional statement, for $\iota \in [d_{out}]$ we see that

$$\begin{aligned}\alpha^L_\cdot(x) &= \sigma(\tilde{\alpha}^L_\cdot(x)), \\ f_\iota(x) = \tilde{\alpha}^{L+1}_\iota(x) &= \beta b^L_\iota + \frac{1}{\sqrt{n_L}}\sum_{\kappa=1}^{n_L} W^L_{\iota\kappa}\alpha^L_\kappa(x).\end{aligned}$$

Then for any $x^{(1)}, \ldots, x^{(m)}$, we have that

$$\begin{bmatrix} f_\iota(x^{(1)}) \\ f_\iota(x^{(2)}) \\ \vdots \\ f_\iota(x^{(m)}) \end{bmatrix} = \beta b^L_\iota \mathbf{1} + \sqrt{n_L}\frac{1}{n_L}\sum_{\kappa=1}^{n_L} W^L_{\iota\kappa} \begin{bmatrix} \alpha^L_\kappa(x^{(1)}) \\ \alpha^L_\kappa(x^{(2)}) \\ \vdots \\ \alpha^L_\kappa(x^{(m)}) \end{bmatrix}.$$

Note the difference between our expression here and that of the base case. We no longer directly have that each summand is a multivariate normal. However that's

where the limit $n_L \to \infty$ comes in. Notice that each $Y_\kappa = W^L_{\iota\kappa} \begin{bmatrix} \alpha^L_\kappa(x^{(1)}) \\ \alpha^L_\kappa(x^{(2)}) \\ \vdots \\ \alpha^L_\kappa(x^{(m)}) \end{bmatrix}$ is an independent draw from the same distribution:

$$W^L_{\iota\kappa} \sim \mathcal{N}(0,1), \quad \begin{bmatrix} \alpha^L_\kappa(x^{(1)}) \\ \alpha^L_\kappa(x^{(2)}) \\ \vdots \\ \alpha^L_\kappa(x^{(m)}) \end{bmatrix} = \sigma\left( \begin{bmatrix} \tilde{\alpha}^L_\kappa(x^{(1)}) \\ \tilde{\alpha}^L_\kappa(x^{(2)}) \\ \vdots \\ \tilde{\alpha}^L_\kappa(x^{(m)}) \end{bmatrix} \right) \sim \sigma\left( \mathcal{N}(0, \Sigma^L(x^{(1)}, \dots, x^{(m)})) \right),$$

so by multivariate CLT we get that

$$\begin{bmatrix} f_\iota(x^{(1)}) \\ f_\iota(x^{(2)}) \\ \vdots \\ f_\iota(x^{(m)}) \end{bmatrix} \overset{n_L \to \infty, d}{\longrightarrow} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\mu} = 0$ (clearly) and $\boldsymbol{\Sigma} = \mathbb{E}[\beta^2 (b^L_\iota)^2 \mathbf{1}\mathbf{1}^\intercal] + \mathbb{E}[W^L_{\iota\cdot} Y_\cdot Y_\cdot^\intercal W^L_{\iota\cdot}]$ where we replaced $\kappa$ with $\cdot$ to denote a generic random variable drawn from the above distribution, which evaluates to

$$\boldsymbol{\Sigma}_{ij} = \beta^2 + \mathbb{E}_{\tilde{\alpha}^L_\cdot \sim \mathcal{GP}(0, \Sigma^L)} \left[ \sigma(\tilde{\alpha}^L_\cdot(x^{(i)}))^\intercal \sigma(\tilde{\alpha}^L_\cdot(x^{(j)})) \right] = \Sigma^{L+1}(x^{(i)}, x^{(j)}),$$

which gives $f_\iota \sim \mathcal{GP}(0, \Sigma^{L+1})$ in the limit as desired.

To show independence, it is also slightly more complicated in this step because we no longer have the fixed input $x$, but rather i.i.d. $\sigma(\tilde{\alpha}^L(x))$. However it can be done by a similar maneuver as above, by applying multivariate CLT to the stacked vector for $\iota \neq \iota'$ via:

(1) We first need to show that in the limit $(f_\iota, f_{\iota'})$ are *jointly Gaussian*. What we've only shown so far is a sort of marginal. The definition of jointly Gaussian processes is analogous to that of jointly Gaussian random variables, but in the processes' settings, it amounts to requiring that the concatenation of finite evaluations of $f_\iota$ and $f_{\iota'}$ are jointly Gaussian. This is easy, by concatenating $(f_{\iota'}(x'^{(1)}), \dots, f_{\iota'}(x'^{(m')}))$ to above and following the same logic.

(2) Once they are jointly Gaussian, to show that they are independent amounts to showing they are uncorrelated. This is a property of Gaussianity. Then it's easy to see that for any $x, x'$, $\mathbb{E}[f_\iota(x) f_{\iota'}(x')] = 0$, because they are linear combinations of separate sets of i.i.d. $\mathcal{N}(0,1)$ weights, namely $\{b^1_a, W^1_{a\cdot}\}$ and $\{b^1_{a'}, W^1_{a'\cdot}\}$, which are themselves independent from $\tilde{\alpha}^L$, so the expectation of product factors into product of expectations, which evaluates to 0.

And thus we are done. □

**Remark 2.7.** Note the convenience of our choice to sequentially send $n_1, \dots, n_{L-1} \to \infty$. Because of this order, the "features" as outputted by the previous layer $\tilde{\alpha}$ follow a well-specified distribution, namely a Gaussian process of an appropriate covariance function. If we were to, say, send $n_1 = \dots = n_{L-1} \to \infty$ simultaneously, the

recursion would not have been nice (e.g. $\tilde{\alpha}^L \sim \frac{1}{\sqrt{n_L}} \sum_{\kappa=1}^{n_L} \cdots$ instead of $\tilde{\alpha}^L \sim \mathcal{GP}$ above), but it is my understanding that it is still doable; see [2].

2.4. **Bayesian inference.** One fun thing one can do is to do Bayesian inference, because when given a training dataset $X = \left(x^{(1)}, \ldots, x^{(m)}\right)$ and a test dataset $X' = \left(x'^{(1)}, \ldots, x'^{(m')}\right)$, and for simplicity only 1 output dimension, the prior of initializing the neural networks parameters as such yields the prior over outputs

$$p(Y, Y') = p(y^{(1)}, \ldots, y^{(m)}, y'^{(1)}, \ldots, y'^{(m')}) = \mathcal{N}(0, \mathbf{\Sigma})$$

where $\mathbf{\Sigma} = \Sigma^L(x^{(1)}, \ldots, x^{(m)}, x'^{(1)}, \ldots, x'^{(m')}) = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XX'} \\ \Sigma_{X'X} & \Sigma_{X'X'} \end{bmatrix}$. However, since we have the ground truth $Y = y^{(1)}, \ldots, y^{(m)}$, this yields back to a standard setting where we can infer

$$Y, Y' \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XX'} \\ \Sigma_{X'X} & \Sigma_{X'X'} \end{bmatrix}\right) \Rightarrow Y'|Y \sim \mathcal{N}(\mu_{Y'|Y}, \Sigma_{Y'|Y})$$

where

$$\begin{aligned} \mu_{Y'|Y} &= \Sigma_{X'X}\Sigma_{XX}^{-1}Y, \\ \Sigma_{Y'|Y} &= \Sigma_{X'X'} - \Sigma_{X'X}\Sigma_{XX}^{-1}\Sigma_{XX'}. \end{aligned}$$

However, I am unsure why one would perform this procedure, because it is not clear that this is a good prior to have.

## 3. Neural tangent kernel

We now shift beyond initialization to inspect what the dynamics of these infinitely wide neural networks through gradient descent (GD) is like. However, before we can do that, it is key to shift our observation to look at the dynamics of the neural network through the function space, on which the cost functional is often convex, instead of parameter space. We take the same setup as above, but now package all weights and bias parameters into $\theta \in \mathbb{R}^P$. Then the neural network is a function $f_\theta(x) : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$, with hidden layers of size $n_1, \ldots, n_{L-1}$.

**Definition 3.1** (Prediction function space)**.** The **prediction function space** is $\mathcal{F} = \left\{f : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}\right\}$. Quite literally, this is the space of all possible functions that accomplish our desired task.

**Definition 3.2** (Realization function)**.** The **realization function** $F^L : \mathbb{R}^P \to \mathcal{F}$ for a particular neural network architecture is the mapping $F^L(\theta) = f_\theta(\cdot)$.

In the supervised learning setting (the NTK paper is a little bit more general on this), assume that we want to learn a true (deterministic) $f^* : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ via a dataset $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right\}$ of $m$ data points where $y^{(i)} = f^*(x^{(i)}) \ \forall \ i \in [m]$. My impression is that this setting is slightly more restrictive than usual as $Y|X$ is not necessarily deterministic, but we brush this aside. The GD update to parameters $\theta$ (post initialization $\theta(0)$ as above) is

$$\theta(t+1) = \theta(t) - \eta \nabla_\theta (C \circ F^L)(\theta(t))$$

where $\eta > 0$ is the learning rate and $C$ quantifies the empirical loss of any prediction function, say the $L^2$ loss

$$C : \mathcal{F} \to \mathbb{R}, \qquad C(f) = \frac{1}{m}\sum_{i=1}^{m} \|f(x^{(i)}) - y^{(i)}\|^2 = \frac{1}{m}\sum_{i=1}^{m} \|f(x^{(i)}) - f^*(x^{(i)})\|^2.$$

Phrased fancily, then $C(f) = \|f - f^*\|_{p_{in}}^2$ where $p_{in}$ is the empirical distribution $p_{in} = \frac{1}{m}\sum_{i=1}^{m} \delta_{x^{(i)}}$ and $\|\cdot\|_{p_{in}}$ is a semi-norm on $\mathcal{F}$ induced by the semi-inner product

$$\langle f, g \rangle_{p_{in}} \coloneqq \mathbb{E}_{x\sim p_{in}}[f(x)^\intercal g(x)] = \frac{1}{m}\sum_{i=1}^{m} f(x^{(i)})^\intercal g(x^{(i)}).$$

We are now curious as to what the training dynamics on the parameter space induces on the functional space. In continuous time, GD on parameter space yields the gradient flow

$$\partial_t \theta(t) = -\nabla_\theta (C \circ F^L)(\theta(t)).$$

For what will soon be clear, let $\mathcal{F}^*$ be (as usual) the dual of $\mathcal{F}$ with the semi-inner product $\langle \cdot, \cdot \rangle_{p_{in}}$. What this then means is that

$$\mathcal{F}^* = \left\{ \mu : \mathcal{F} \to \mathbb{R}, \mu = \langle d, \cdot \rangle_{p_{in}} \text{ for some } d \in \mathcal{F} \right\},$$

because for $\mu$ to be continuous it has to have $\|f\|_{p_{in}} = 0 \Rightarrow \mu(f) = 0$, which means $\mu(f)$ can only depend on $f$ on $p_{in}$. Linearity of $\mu$ then implies $\mu(f) = \frac{1}{m}\sum_{i=1}^{m} d(x^{(i)})^\intercal f(x^{(i)}) = \langle d, f \rangle_{p_{in}}$ for some appropriate $d \in \mathcal{F}$. Note that the choice of $d$ does not matter outside of $p_{in}$. A more structural argument is that $\mathcal{H} = \mathcal{F}/\{f : \|f\|_{p_{in}} = 0\}$, quotiented by the null space, is a finite-dimensional ($m \times d_{out}$) inner product space, hence a Hilbert space, and $\mathcal{F}^* \cong \mathcal{H}^*$, so by Riesz any $\mu \in \mathcal{F}^*$ can be identified with a $[d] \in \mathcal{H}$. In this view it is also explicit why only values of $d$ on $p_{in}$ matter, because it is $[d] \in \mathcal{H}$ that matters. The above, perhaps unnecessary spiel was simply to massage and convince the reader (and the author) that these definitions are mathematically tight.

Why we want to set up such a space is that since the cost functional $C$ only depends on values of $f \in \mathcal{F}$ at data points, the functional derivative of $C$ with respect to any $f_0 \in \mathcal{F}$ can be viewed as $\partial_f C|_{f_0} \in \mathcal{F}^*$, hence identifiable with some $d|_{f_0} \in \mathcal{F}$, i.e. $\partial_f C|_{f_0}(\cdot) = \langle d|_{f_0}, \cdot \rangle_{p_{in}}$.

By 2 applications of chain rule, GD on the parameter space hence induces the dynamics on the function space

$$\begin{aligned}
\partial_t f_{\theta(t)}(x) &= \sum_{p=1}^{P} \partial_{\theta_p} f_{\theta(t)}(x) \cdot \partial_t \theta_p(t) && \left(\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\mathrm{d}f}{\mathrm{d}\theta}\frac{\mathrm{d}\theta}{\mathrm{d}t}\right) \\
&= \sum_{p=1}^{P} \partial_{\theta_p} f_{\theta(t)}(x)(-\partial_{\theta_p}(C \circ f_{\theta(t)})) && \left(\frac{\mathrm{d}\theta}{\mathrm{d}t} = \nabla_\theta (C \circ f)\right) \\
&= -\sum_{p=1}^{P} \partial_{\theta_p} f_{\theta(t)}(x) \left\langle d|_{f_{\theta(t)}}, \partial_{\theta_p} f_{\theta(t)} \right\rangle_{p_{in}} && \left(\frac{\mathrm{d}(C \circ f)}{\mathrm{d}\theta} = \frac{\mathrm{d}C}{\mathrm{d}f}\frac{\mathrm{d}f}{\mathrm{d}\theta}\right) \\
&= -\frac{1}{m}\sum_{i=1}^{m} \Theta(x, x^{(i)}) d|_{f_{\theta(t)}}(x^{(i)}) && \text{(grouping terms)}
\end{aligned}$$

where $\Theta(x, x^{(i)}) = \sum_{p=1}^{P} \partial_{\theta_p} f_{\theta(t)}(x) \partial_{\theta_p} f_{\theta(t)}(x^{(i)})^\intercal = (\nabla_\theta f_{\theta(t)}(x))^\intercal \nabla_\theta f_{\theta(t)}(x^{(i)})$. More generally, $\Theta(x, x') \coloneqq (\nabla_\theta f_{\theta(t)}(x))^\intercal \nabla_\theta f_{\theta(t)}(x')$ is the **neural tangent kernel** (NTK). The functional dynamics for each index $\iota$ of $f_{\theta(t)}$ is a somewhat neater expression, which is

$$\partial_t f_{\iota,\theta(t)}(x) = -\frac{1}{m}\sum_{i=1}^{m} \Theta_\iota(x, x^{(i)}) d|_{f_{\theta(t)}}(x^{(i)}) = -\left\langle \Theta_\iota(x, \cdot), d|_{f_{\theta(t)}} \right\rangle_{p_{in}}. \tag{3.3}$$

3.1. **Kernels 101.** The jumble above might not make much intuitive sense, but we offer some interpretation here, and justification as to why $\Theta$ above is called a kernel. Note that we have not made any statement about the infinite limit, so the above applies to any fully-connected feedforward neural network (and more).

Let us consider the very easy problem of finding the optimal weight $w^* \in \mathbb{R}^d$ for linear regression to predict $y \in \mathbb{R}$ from $x \in \mathbb{R}^{d_{in}}$, with the feature map $\phi : \mathbb{R}^{d_{in}} \to \mathbb{R}^d$, via minimizing the empirical $L^2$ loss. Then the prediction function is $f_w : \mathbb{R}^d \to \mathbb{R}, x \mapsto w^\intercal \phi(x)$, and we are trying to find

$$w^* = \arg\min_w \frac{1}{m}\|\phi(X)w - y\|_2^2 = \arg\min_w \frac{1}{m}\sum_{i=1}^{m}(w^\intercal \phi(x^{(i)}) - y^{(i)})^2,$$

where $\phi(X) \in \mathbb{R}^{m \times d}$ is the design matrix. However the expression above can be ill-formed. When $\phi(X)^\intercal \phi(X)$ is invertible, there is a unique solution whose closed-form expression is $w^* = (\phi(X)^\intercal \phi(X))^{-1}\phi(X)^\intercal y$. When it is not, it is because $\phi(X)$ is underdetermined; hence the minimizer is not unique, thus we make the intuitive choice to define $w^*$ with minimum $L^2$ norm, whose "closed-form" expression is $\phi(X)^\dagger y$. Regardless, we can make the definition of $w^*$ more precise:

$$w^* \coloneqq \arg\min_w \|w\|_2 \text{ such that } \|\phi(X)w - y\|_2^2 \text{ is minimized.}$$

**Proposition 3.4.** *In both cases above, $w^*$ is in the span of the training points. In other words, $w^* \in \left\{\phi(x^{(i)}) : i \in [m]\right\} = \mathrm{im}(\phi(X)^\intercal)$.*

The above is not really Representer Theorem, but the idea is very much the same, because one can always decompose the weight/feature space into $\mathrm{im}(\phi(X)^\intercal) \oplus \mathrm{im}(\phi(X)^\intercal)^\perp$, and any component from the orthogonal subspace would have no effect on prediction (due to orthogonality with the data points), while increasing

the norm, and hence $w^*$ should have no component from the orthogonal subspace and only lie on $\mathrm{im}(\phi(X)^\intercal)$.

**Corollary 3.5.** *It suffices to find* $\left\{\alpha^{*(i)} \in \mathbb{R} : i \in [m]\right\}$ *for* $w^* = \sum_{i=1}^m \alpha^{*(i)}\phi(x^{(i)})$*, i.e., the vector* $\alpha^* \in \mathbb{R}^m$ *such that* $w^* = \phi(X)^\intercal\alpha^*$*.*

To relate back to the neural network setting which is trained via GD, one can also run GD on this problem and be guaranteed convergence because the loss function is convex with respect to $w$. It is then reasonable with initialize anywhere, and in particular at $w(0) = 0$ corresponding to initializing the dual variable $\alpha(0) = 0$. The GD updates then yield

$$w(t+1) = w(t) - \frac{2\eta}{m}\sum_{i=1}^m (w(t)^\intercal\phi(x^{(i)}) - y^{(i)})\phi(x^{(i)})$$

which corresponds to updating

(3.6)
$$\alpha(t+1)^{(i)} = \alpha(t)^{(i)} - \frac{2\eta}{m}\sum_{i=1}^m (w(t)^\intercal\phi(x^{(i)}) - y^{(i)}), \quad w(t+1) = \sum_{i=1}^m \alpha(t+1)^{(i)}\phi(x^{(i)}).$$

The first observation is that through the optimization process $w(t) \in \mathrm{im}(\phi(X)^\intercal)$, and since the optimization problem is convex, it would converge to a minimizer in $\mathrm{im}(\phi(X)^\intercal)$. It is then easy to see that that minimizer is exactly the minimum-norm $w^*$.

The second, bigger observation is that one actually only uses the inner products of the (featurized) inputs throughout the GD process for $\alpha$, which involved the terms

$$w(t)^\intercal\phi(x^{(i)}) - y^{(i)} = -y^{(i)} + \sum_{j=1}^m \alpha(t)^{(j)}\left\langle \phi(x^{(i)}), \phi(x^{(j)})\right\rangle,$$

as well as during inference under this representation, because

$$f_{w(t)}(x) = w(t)^\intercal\phi(x) = \sum_{j=1}^m \alpha(t)^{(j)}\left\langle \phi(x), \phi(x^{(j)})\right\rangle$$

for the same reason. Therefore, we can circumvent explicitly defining the feature map $\phi$ by implicitly defining it via the specification of this inner product, a **kernel** $K(x, x') = \langle\phi(x), \phi(x')\rangle$, which makes the gradient updates in the dual variables

$$\alpha(t+1)^{(i)} - \alpha(t)^{(i)} = -\frac{2\eta}{m}\left[-y^{(i)} + \sum_{i=1}^m \alpha(t)^{(j)}K(x^{(i)}, x^{(j)})\right] \tag{3.7}$$

and inference at any iterate

$$f_{w(t)}(x) = w(t)^\intercal\phi(x) = \sum_{i=1}^m \alpha(t)^{(j)}K(x, x^{(i)}). \tag{3.8}$$

**Remark 3.9.** The intuition of what $K$ signifies is perhaps clearest from the inference equation. Intuitively, $K(x, x')$ denotes how *similar* $x$ and $x'$ are, as suggested by the feature map $\phi$ (it is literally the dot product of $\phi(x)$ and $\phi(x')$) — the prediction on a test point $x$ is a linear combination of values $\alpha$ assigned to training points, weighted by how similar $x$ is to each training point. This **kernel trick** have two implications: (1) Bypassing the dot product saves tremendous computation,

and (2) by picking clever kernels like the Gaussian radial basis function (RBF) $K(x, x') = e^{-\|x-x'\|^2/2}$, it is analogous to using an infinite dimensional feature map $\phi$ (which can be seen via the power series representation of the above), which would have been impossible via an explicit definition of $\phi$.

It is clear that any $K(x, x')$ as defined in this manner is symmetric and positive semidefinite (PSD), concretely in the sense that for any $\left\{x^{(i)} : i \in [m]\right\}$, the **Gram matrix** $\mathbf{K} = [K(x^{(i)}, x^{(j)})]_{i,j\in[m]}$ is symmetric and PSD. The following subsection then says that this is the sufficient condition; that symmetric, PSD $K$ corresponds to some feature map.

3.2. **Reproducing kernel Hilbert spaces.** Let $\mathcal{X}$ be a generic input space, which can have no structure whatsoever. So far we've used $\mathcal{X} = \mathbb{R}^{d_{in}}$.

**Definition 3.10** (Reproducing kernel)**.** A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **reproducing kernel** if for any $x^{(1)}, \ldots, x^{(m)} \in \mathcal{X}$ we get that the Gram matrix $\mathbf{K} = [K(x^{(i)}, x^{(j)})]_{i,j\in[m]} \in \mathbb{R}^{m\times m}$ is symmetric and PSD.

Then the concept of a reproducing kernel Hilbert space (RKHS) corresponding to a reproducing kernel $K$ makes concrete the space that $K$'s corresponding feature map maps into.

**Definition 3.11** (RKHS)**.** A Hilbert space $\mathcal{H}$ of real-valued functions from $\mathcal{X} \to \mathbb{R}$ (equipped with pointwise addition and scalar multiplication) is an **RKHS** if the evaluation functional $ev_x : \mathcal{H} \to \mathbb{R}, f \mapsto f(x)$ is a continuous operator on $\mathcal{H}$.

By Riesz, that implies for each $x$, there exists a unique $K_{\mathcal{H},x} \in \mathcal{H}$ such that

$$ev_x(f) = \langle K_{\mathcal{H},x}, f\rangle_{\mathcal{H}},$$

which induces a unique $K_{\mathcal{H}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined via $K_{\mathcal{H}}(x, x') = K_{\mathcal{H},x}(x')$. An interesting implication is that since $K_{\mathcal{H},x} \in \mathcal{H}$ itself for any $x \in \mathcal{X}$, it follows that for any $x, x' \in \mathcal{X}$,

$$K_{\mathcal{H}}(x, x') \overset{\text{defined above}}{=} K_{\mathcal{H},x}(x') \overset{\text{evaluation}}{=} ev_{x'}(K_{\mathcal{H},x}) \overset{\text{Riesz}}{=} \langle K_{\mathcal{H},x'}, K_{\mathcal{H},x}\rangle_{\mathcal{H}}$$

which is symmetric, so $K_{\mathcal{H}}(x, x') = K_{\mathcal{H}}(x', x)$, and also PSD, which one can show by brute force. However, those properties are obvious after realizing the apparent feature map $\phi$ here, namely $\phi : \mathcal{X} \to \mathcal{H}, \phi(x) = K_{\mathcal{H},x}(\cdot) = K_{\mathcal{H}}(x, \cdot)$, which makes $K_{\mathcal{H}}(x, x') = \langle\phi(x), \phi(x')\rangle_{\mathcal{H}}$.

**Proposition 3.12.** *Let $\mathcal{H}$ be an RKHS, then (the uniquely constructed) $K_{\mathcal{H}}$ is a reproducing kernel, and its corresponding feature map is $\phi(x) = K_{\mathcal{H},x}(\cdot) \in \mathcal{H}$.*

The following theorem is then a converse.

**Theorem 3.13** (Moore-Aronszajn)**.** *Let $K$ be a reproducing kernel. Then there exists a RKHS $\mathcal{H}$ such that $K_{\mathcal{H}} = K$. In fact $\mathcal{H}$ is unique up to isometric isomorphism.*

The upshot of this theorem is that we have a correspondence between reproducing kernels and RKHS, The **reproducing** in the name comes from the fact that $K_{\mathcal{H},x}$ "reproduces" $f(x)$ via $\langle K_{\mathcal{H},x}, f\rangle_{\mathcal{H}}$.

*Proof.* I only include the proof here because there seems to exist some ambiguity as to whether $K$ has to be positive definite, or simply being positive semidefinite is enough. The answer is that PSD is sufficient.

Define $\mathcal{H}_0 = \operatorname{span}\{K(x,\cdot) : x \in \mathcal{X}\}$. It is a vector space over $\mathbb{R}$ with pointwise addition and pointwise scalar multiplication. Claim that

$$\left\langle \sum_{i=1}^m a_i K(x^{(i)},\cdot), \sum_{j=1}^m b_j K(x^{(j)},\cdot) \right\rangle_{\mathcal{H}_0} := \sum_{i,j=1}^m a_i b_j K(x^{(i)}, x^{(j)})$$

is an inner product. Once we manage to show that, we just need to complete $\mathcal{H}_0$ to get $\mathcal{H}$, and inherit the inner product.

Symmetry and linearity are easy. The only slightly non-trivial property is positive definiteness, so take any $f = \sum_{i=1}^m a_i K(x^{(i)},\cdot)$ such that $\langle f, f\rangle_{\mathcal{H}_0} = 0$.

The obvious thing to try here is to let $\mathbf{K} = [K(x^{(i)}, x^{(j)})]_{i,j\in[n]}$ as usual, then $\langle f, f\rangle_{\mathcal{H}_0} = 0$ implies that $a^\intercal \mathbf{K} a = 0$, but that does not necessarily imply $a = 0$. However, we should take advantage of the reproducing-ness of this inner product. Take any $x \in \mathcal{X}$ then

$$\begin{aligned} f(x) &= \sum_{i=1}^m a_i K(x^{(i)}, x) && \text{(plug in for } f\text{)} \\ &= \langle f, K(x,\cdot)\rangle_{\mathcal{H}_0} && \big(\text{definition of } \langle\cdot,\cdot\rangle_{\mathcal{H}_0}\big) \end{aligned}$$

so indeed it is reproducing (at least with $\langle\cdot,\cdot\rangle_{\mathcal{H}_0}$). That implies

$$f(x)^2 = \big(\langle f, K(x,\cdot)\rangle_{\mathcal{H}_0}\big)^2 \le \langle f, f\rangle_{\mathcal{H}_0} \langle K(x,\cdot), K(x,\cdot)\rangle_{\mathcal{H}_0} = 0 \Rightarrow f(x) = 0,$$

where it was valid to use Cauchy-Schwarz.

$K$ is reproducing with $\langle\cdot,\cdot\rangle_{\mathcal{H}_0}$, so it is also reproducing with $\langle\cdot,\cdot\rangle_{\mathcal{H}}$. By Riesz, it is unique, so indeed $K_{\mathcal{H}} = K$. We have constructed the RKHS $\mathcal{H}$ that corresponds with $K$. □

3.3. **Linear regression is... a neural network.** Other than (3.7), which shows kernelized GD updates in the dual variables, and (3.8), which shows kernelized inference, we can write down another equation that is a combination of those two (sort of), which is the kernelized inference update, i.e., GD in the function space:

$$\begin{aligned} f_{w(t+1)}(x) - f_{w(t)}(x) &= w(t+1)^\intercal \phi(x) - w(t)^\intercal \phi(x) \\ &\overset{(3.8)}{=} \sum_{i=1}^m (\alpha(t+1)^{(i)} - \alpha(t)^{(i)}) K(x, x^{(i)}) \\ &\overset{(3.6)}{=} -\eta \frac{1}{m} \sum_{i=1}^m K(x, x^{(i)}) \left[2(w(t)^\intercal \phi(x^{(i)}) - y^{(i)})\right] \end{aligned}$$

which in the continuum is

$$\partial_t f_{w(t)}(x) = -\frac{1}{m} \sum_{i=1}^m K(x, x^{(i)}) \left[2(w(t)^\intercal \phi(x^{(i)}) - y^{(i)})\right]. \tag{3.14}$$

The apt reader would, as I should have a lot sooner[2], realize that this is exactly the NTK equation of GD in function space for this particular degenerate neural network, because linear regression is just a neural network with depth $L = 1$, and $L - 1 = 0$ hidden layers. I recall (3.3) here for ease of comparison:

$$\partial_t f_{\iota,\theta(t)}(x) = -\frac{1}{m}\sum_{i=1}^{m} \Theta_\iota(x, x^{(i)}) d|_{f_{\theta(t)}}(x^{(i)}) = -\left\langle \Theta_\iota(x,\cdot), d|_{f_{\theta(t)}} \right\rangle_{p_{in}}.$$

Conforming to this notation, we can also write

$$\partial_t f_{w(t)}(x) = -\left\langle K(x,\cdot), 2(w(t)^\intercal \phi(\cdot) - f^*(\cdot)) \right\rangle_{p_{in}},$$

where we aptly notice that if we view linear regression as a neural network with $\theta \equiv w$, then

(1) by definition $\Theta(x, x') = \nabla_\theta f_\theta(x)^\intercal \nabla_\theta f_\theta(x')$, and combined with $f_w(x) = w^\intercal \phi(x) \Rightarrow \nabla_w f_w(x) = \phi(x)$, it yields $\Theta(x, x') = \phi(x)^\intercal \phi(x')$, which is exactly $K(x, x')$,

(2) and $2(w(t)^\intercal \phi(\cdot) - f^*(\cdot))$ is exactly the functional derivative of the $L^2$ loss, so it corresponds to the $d|_{f_{\theta(t)}}$ term,

so everything matches up as expected. Alas, we have done the computation of another base case.

**Remark 3.15.** The intuition of $K$ denoting similarity once again comes in handy here. Not only does it apply to the inference (per Remark 3.9), but it also applies similarly for inference updates, i.e. GD in function space — the functional gradient is a weighted update with weights proportional to how similar $x$ is to every training data point.

**Remark 3.16.** It is now clear from all above why $\Theta(x, x') = \nabla_{\theta(t)} f_\theta(x)^\intercal \nabla_{\theta(t)} f_\theta(x')$ is called the **neural tangent kernel**. It is the **kernel** induced by the feature map $\phi : x \mapsto \nabla_\theta f_{\theta(t)}(x)$, which is the **tangent** vector of $f_\theta$ at $\theta(t)$ when evaluated with $x$. It is **neural**, well, because $f_\theta$ is a neural network.

**Remark 3.17.** Despite the above, it is misleading to only frame this kernel in the fully-connected neural networks setting. In fact, nothing we have written is dependent on the structure of the neural network. Thus a similar viewpoint can be taken for other architectures, e.g. convolutional neural networks, transformers, etc. Well, it doesn't have to be a neural network either — just about any $f_\theta$ (maybe sufficiently nice) would work.

3.4. **Kernel gradient descent and convergence.** The gist of all above is that we can view GD on function space as linear regression with a very particular feature map, which is encoded in a very particular kernel, the NTK. To align notations with [1], we define the **kernel gradient** $\nabla_K C|_{f_0} \in \mathcal{F}$ for a multi-dimensional kernel[3] $K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{d_{out} \times d_{out}}$ to be

$$\nabla_K C|_{f_0}(x) = \frac{1}{m}\sum_{i=1}^{m} K(x, x_i) d|_{f_0}(x^{(i)}),$$

[2]I initially only wanted to include the subsection on linear regression above to introduce ideas of kernels and RKHS, but it was a happy accident.

[3]Defined analogously, now $K(x, x') = K(x', x)^\intercal$.

i.e. functional gradient updates weighted by similarities to training points as judged by $K$, and the **kernel gradient descent with respect to** $K$ procedure to be

$$\partial_t f(t) = -\nabla_K C|_{f(t)}.$$

This changes nothing in our discussion, other than the fact that GD on parameter space induces kernel GD with the particular kernel $\Theta$. However, with this notation we can talk about when running kernel gradient descent with any $K$ would achieve convergence. Indeed, the cost functional would evolve as

$$\begin{aligned}
\partial_t C|_{f_{\theta(t)}} &= \partial_f C|_{f_{\theta(t)}}(\partial_t f_{\theta(t)}) && \text{(Chain rule)}\\
&= \left\langle d|_{f_{\theta(t)}}, -\nabla_K C|_{f_{\theta(t)}} \right\rangle_{p_{in}}\\
&= \left\langle d|_{f_{\theta(t)}}, -\frac{1}{m}\sum_{i=1}^m K(\cdot, x^{(i)}) d|_{f_{\theta(t)}}(x^{(i)}) \right\rangle_{p_{in}} && \text{(Kernel gradient)}\\
&= \frac{1}{m}\sum_{j=1}^m d|_{f_{\theta(t)}}(x^{(j)})^\intercal \left[\frac{1}{m}\sum_{i=1}^m K(x^{(j)}, x^{(i)}) d|_{f_{\theta(t)}}(x^{(i)})\right]\\
&= \mathbb{E}_{x,x'\sim p_{in}}\left[d|_{f_{\theta(t)}}(x)^\intercal K(x,x') d|_{f_{\theta(t)}}(x')\right],
\end{aligned}$$

which suggests defining the semi-inner product

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

with a corresponding norm. Then the above expression yields

$$\partial_t C|_{f_{\theta(t)}} = -\|d|_{f_{\theta(t)}}\|_K,$$

Recall that $d|_{f_{\theta(t)}}$ is the derivative of the cost with respect to the predicted values. Thus, if kernel $K$ is positive definite with respect to $\|\cdot\|_{p_{in}}$, that is $\|p_{in}\| > 0 \Rightarrow \|f\|_K > 0$, then the cost is strictly decreasing over time, except at points with $\|d|_{f_{\theta(t)}}\|_{p_{in}} = 0$, i.e., critical points of $C$ — thus convergence to a critical point is guaranteed. If $C$ is further convex and bounded from below (say $L^2$ loss), all critical points are global minima, so $f_{\theta(t)}$ converges to one as $t \to \infty$.

The apt reader would notice that the convergence discussion above only works if (1) $K$ is indeed positive definite with respect to $\|\cdot\|_{p_{in}}$, which is somewhat achievable, but more crucially (2) $K$ is fixed across time. As $\theta$ changes over time, $\Theta(x,x') = (\nabla_\theta f_{\theta(t)}(x))^\intercal \nabla_\theta f_{\theta(t)}(x')$ should also change over time, making the results above inapplicable. However, this is exactly what happens when we run GD on infinitely wide networks.

## 4. Infinitely wide neural networks through gradient descent

4.1. **Optimization.** We'll now try to see what happens to these kernels as $n_1, \dots, n_{L-1} \to \infty$. Since we'll talk about the NTK at different depths, we'll refer to $\Theta^L$ as the NTK corresponding to networks of $L-1$ hidden layers and 1 output layer. In particular, we saw the explicit computation of $\Theta^1$ above.

**Theorem 4.1** ([1], Theorem 1)**.** *At initialization, for a network of depth $L$, in the limit $n_1, \dots, n_{L-1} \to \infty$ sequentially, the NTK $\Theta^L$ converges in probability to a deterministic limiting kernel*

$$\Theta^L \to \Theta^L_\infty \otimes Id_{d_{out}}.$$

*The scalar kernel* $\Theta_\infty^L : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ *is defined recursively by*

$$\Theta_\infty^1(x, x') = \Sigma^1(x, x'),$$
$$\Theta_\infty^{l+1}(x, x') = \Theta_\infty^L(x, x')\dot{\Sigma}^{L+1}(x, x') + \Sigma^{L+1}(x, x'),$$

*where*

$$\dot{\Sigma}^{L+1}(x, x') = \mathbb{E}_{f\sim\mathcal{N}(0,\Sigma^L)}\left[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))\right].$$

$\dot{\sigma}$ *denotes the derivative of* $\sigma$*, and* $\Sigma^l$ *is the covariance function in the previous section.*

**Remark 4.2.** In English, the deterministic kernel is

$$(\Theta_\infty^L \otimes Id_{d_{out}})(x, x') = \Theta_\infty^L(x, x')I_{d_{out}} \in \mathbb{R}^{d_{out}\times d_{out}},$$

and the convergence in probability above means pointwise convergence in probability, i.e. every $x, x' \in \mathbb{R}^{d_{in}}$.

*Proof.* We prove again by induction on $L$. We have actually shown that this kernel is deterministic for the base case $L = 1$, which is just linear regression. Due to the $\frac{1}{\sqrt{n_0}}$ factor in the parameterization of the weights, the presence of the bias term, and that the feature map is just identity,

$$\Phi^1(x, x') = \frac{1}{n_0}x^\intercal x' + \beta^2$$

deterministically, which is also $\Sigma^1(x, x')$. The inductive step from depth $L$ to depth $L+1$ is then more applications of chain rule, where we can essentially see the term $\Theta_\infty^L(x, x')\dot{\Sigma}^{L+1}(x, x')$ as the chain rule over parameters in the first $L$ layers, which has as the activation-related term $\dot{\Sigma}^{L+1}$, while the term $\Sigma^{L+1}(x, x')$ is for the last, newly added layer. □

**Remark 4.3.** In English, the theorem above suggests that at initialization of networks with infinitely wide hidden layers, the NTK is effectively deterministic. Theorem 2 of [1] then says that through the training process, as $n_1, \ldots, n_{L-1} \to \infty$ sequentially, the NTK converges uniformly for $t \in [0, T]$ to the deterministic kernel as initialized, namely $\Theta_\infty^L \otimes Id_{d_{out}}$. The proof is rather tedious, but is "just chain rule."

**Corollary 4.4.** *The dynamics of* $f_\theta$ *of depth* $L$ *with infinitely wide hidden layers as induced by GD on parameter space is kernel gradient descent with respect to a fixed, deterministic kernel*

$$\partial_t f_{\theta(t)} = -\nabla_{\Theta_\infty^L \otimes Id_{d_{out}}} C|_{f_\theta(t)}.$$

Combined with Proposition 2 from [1], which says that for $L \geq 2$, $\Theta_\infty^L \otimes Id_{d_{out}}$ is positive definite when restricted to the unit sphere $\mathbb{S}^{n_0-1}$ (which is not super restrictive, because for high-dimensional data, centered data points often have roughly the same norm), the above corollary implies that indeed these GD-trained networks converge to a global minimum of the empirical risk. We call these "interpolation" solutions (if the cost functional $C$ actually reflects this, but of course it should).

4.2. **Generalization.** Once we have resolved the question of ERM optimization, generalization must be shown as well. Indeed, not all interpolating solutions are created equal, especially in this case where the dynamics is that of kernel gradient descent.

Per our discussion on RKHS, the limiting kernel $\Theta = \Theta^L = \Theta^L_\infty \otimes Id_{d_{out}}$ has a corresponding RKHS $\mathcal{H}_\Theta$ with elements of type $\sum_{i=1}^m a_i \Theta(x^{(i)}, \cdot)$, and inner product

$$\left\langle \sum_{i=1}^m a_i \Theta(x^{(i)}, \cdot), \sum_{j=1}^m b_j \Theta(x^{(i)}, \cdot) \right\rangle_{\mathcal{H}_\Theta} = \sum_{i,j=1}^m a_i b_j \Theta(x^{(i)}, x^{(j)}) = a^\intercal \mathbf{\Theta} b$$

and norm

$$\left\| \sum_{i=1}^m a_i \Theta(x^{(i)}, \cdot) \right\|_{\mathcal{H}_\Theta} = a^\intercal \mathbf{\Theta} a.$$

**Proposition 4.5.** *Kernel GD converges to the solution*

$$f^*_\infty = \|f\|_{\mathcal{H}_\Theta} \text{ such that } \frac{1}{m} \sum_{i=1}^m \|f(x^{(i)}) - y^{(i)}\|^2 \text{ is minimized.}$$

*Proof.* This is the same idea as Proposition 3.4. □

**Corollary 4.6.** *The solution that GD on infinitely wide neural networks converges to is the simplest interpolating solution through the lens of $\|\cdot\|_{\mathcal{H}_\Theta}$. With such complexity control, one has hope that it has good generalization.*

## 5. What it means

So, I've said a lot. What are some of my takeaways from NTK to understand neural networks?

### 5.1. **The good.**

(1) The NTK point of view, i.e. equivalence of running GD in parameter space with running kernel GD in function space is interesting, and generalizable to a lot of other architectures.

(2) From essentially just using chain rule and LLN/CLT, we can get a pretty clean description of these networks.

(3) The implicit bias of kernel GD (minimizing corresponding RKHS norm) shows that an interpolating solution can have generalization.

### 5.2. **The bad.**

(1) Despite the clean theoretical framework, NTK doesn't exactly explain how finite-sized networks with non-infinitesimal step sizes still manage to achieve empirically demonstrated good generalization.

(2) With dynamics in the limit case being kernel GD with respect to a fixed kernel, that essentially means that the network is not learning new features, and is instead simply doing linear learning on this fixed feature map. This is not really compatible with the view (of mine) that these neural networks do gradually learn some sort of features and are not static.

5.3. **The awesome.**

(1) I got the opportunity to learn more deeply and write about this subject! This is a final report I wrote for the course MATH 35607: Introduction to Machine Learning at the University of Chicago in Autumn 2025. I would like to thank Professor Danny Calegari and Professor Amie Wilkinson, who gave us an abundance of tea breaks and were amazing instructors throughout the quarter.

## Acknowledgment

See above.

## References

[1] A. Jacot, F. Gabriel, and C. Hongler, "Neural Tangent Kernel: Convergence and Generalization in Neural Networks," in *Advances in Neural Information Processing Systems* (NeurIPS), 2018.

[2] A. G. de G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani, "Gaussian Process Behaviour in Wide Deep Neural Networks," in *International Conference on Learning Representations* (ICLR), 2018.