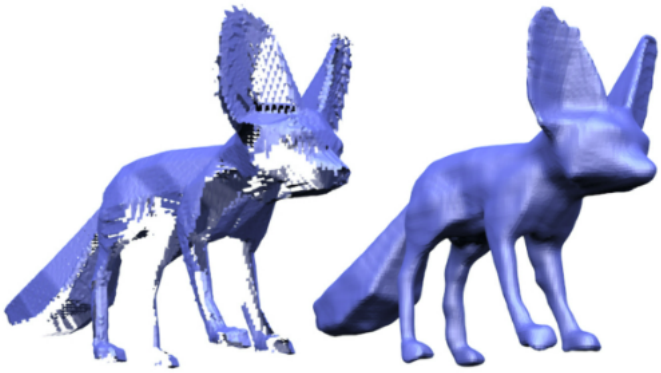
Notes on The Neural Tangent Kernel

A beginners' guide

Simone Maria Giancola*†

†Bocconi University, Milan

January 31, 2023



Abstract

The following document is an exploration of the results of [JGH20], written to understand better the content of the claims. The presentation is at this link. It is not an extension, but rather an expansion of some of the elements needed for a less experienced reader. As this production is done in fulfillment of a semester exam for an Machine Learning course, it does not cover all of the content. The focus is on the formulation and properties of the Neural Tangent Kernel, and all the theoretical results needed to understand them. The works cited are in line with those of the authors, with some additional resources that I found helpful. Given the breadth of the subject, some of the content is left for future studies.
Section I discusses the peculiarities of the setting chosen by the authors. Sections II, III introduce the Neural Tangent Kernel and show the conditions for it to be well formulated. Section IV is a collection of interesting interpretations of the consequences of the NTK regime, partly empirically checked in Section V. Section VI comments on the results found and gives an overview of some of the contributions that stemmed from the original work.
In Appendix A it is possible to find more details about the mathematical objects used throughout the document. The image is a cutout of the wonderful cover of [CPW21].

*simonegiancola09@gmail.com

Contents

List of Figures

I Setting

This Section is devoted to a presentation of the assumptions and structure of the problem. To give context, a brief comment on theoretical results in the field is provided, almost completely inspired by the NTK paper [JGH20]. Subsequenty, the architecture of interest for the document is thouroughly explained, in both its parametric and functional formulation. The Kernel trick is presented as a way to solve optimization problems by transforming features, with its advantages and limitations. On this line of thought, a constructive explanation of multidimensional Kernels is provided. With a final example of kernel gradient descent dynamics arising from an approximation problem.

Main Sources This document is intended for someone that wishes to read a more detailed explanation of the paper that introduced the Neural Tangent Kernel [JGH20]. It is an exploration rather than an expansion, targeted for less experienced readers such as myself. To achieve this objective I have used complementary sources with different formats, which could be summarized into three main categories:

- lectures of ML theory courses [Soh20; Ten22a; Ten22b]
- researcher's blogs [Vad19; Hus20; Wal21; Wen22]
- comments to the calculations by Yilan Chen and Mateusz Mroczka and Benedikt Petko

The notation is the one chosen by the original authors. Content exposition is inspired as well by the order of the reference paper but might have some detours on important aspects. The Appendix presents additional basic results with general references if someone is interested in the theoretical building blocks of this subject. Proofs are in most cases just an expansion of the steps of the ones proposed. If a proof is not reported, a reference is given for the sake of completeness. If a reference is not provided, it means that it is a result easy enough to find with a web search.

I.1 Introduction

Artificial Neural Networks (ANNs) are widely studied from a theoretical point of view, as they prove to be instrumental for understanding the current landscape of learning methods. Despite their practical success, there are a number of open questions in the field. The main issue is understanding convergence, in which terms, and with which properties.
In this document, a nice connection with Kernel methods will be drawn, thanks to the interesting work by the authors of the main source [JGH20], which provides a brief overview of the present situation. Despite being certainly non exhaustive, it is part of the thought process that lead to their results. Below, in a bullet point fashion, the discussion is summarized.

- NNs are capable of approximating any function provided that their size is sufficient [HSW89; Les+93]
- the instrinsic nonlinearity in the parameters makes the problem hard, with an optimization landscape which is highly non-convex. Gradient descent on the parameters gets stuck at the multiple saddle points (local minima) [Dau+14]

- under strong assumptions, it was shown that despite the broken dynamics, the quality of local minima is satisfactory to some extent (i.e. the few bad local minimas results) [Pas+14; Cho+15; PB17]
- there have been recent developments in the geometry of the loss landscape [KAA19] and diverging hidden neurons dynamics [MMN18]
- generalization properties in the over-parametrization regime [Sag+18] have not been explained yet. Neural Networks are well performing even under training with random labels and testing on real data [Zha+17].

The connection with kernel methods is instead justified by the following motivations:

- kernel methods are higher dimensional projections of data that remain linear in the parameters, leaving the optimization problem simple
- they present the same surprising performances on real data after random training [BMM18]
- they are a well understood field, and have already been connected with infinite-width ANNs, which end up being Gaussian processes with kernel dynamics [Nea96; DFS17; Mat17; Lee+18; Mat+18]
- results are close to ANNs trained with gradient descent [CS09; Lee+18]

For further considerations, another introductory overview is proposed in [CB18]. On the a slightly different Geometric-oriented field, but with a highly detailed reference list, a suggestion is also the GDL proto book [Bro+21].

I.2 Artificial Neural Networks

We first clearly state the learning setting considered.

Definition I.2.1 (Experimental Setup)**.** *We have a finite dataset* $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$*, where* $x_i \in \mathscr{X} \subseteq \mathbb{R}^{n_0} \forall i$ *and* $y_i \in \mathscr{Y} \subseteq \mathbb{R}^{n_L} \forall i$*. We could equivalently store the all features and targets into matrices:*

$$\boldsymbol{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \in \mathbb{R}^{N \times n_0} \quad \vec{\boldsymbol{y}} = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} \in \mathbb{R}^{N \times n_L}$$

Where the latter is expressed in vector notation since it is often the case that $n_L = 1$*. The features matrix is assumed to be an* N *sized iid sample from an input distribution* p^{in} *over* $\mathscr{X}$*.*

Definition I.2.2 (Artificial Neural Network Architecture, ANN)**.** *One of the simplest non-linear learning models is the ANN. Its construction is that of a fully connected Neural Network with:*

- *layers* $\ell \in \{0, \dots, L\}$ *where the* 0^{th} *layer is the input, and the* L^{th} *layer is the output*
- *neurons* $n_0, \dots, n_L$ *at each layer*
- *nonlinearities* $\sigma : \mathbb{R} \to \mathbb{R}$*, applied element-wise*

The trainable parameters for each layer but the last $\ell \in \{0, \dots, L-1\}$ *are:*

- *weights* $W^{(\ell)} \in \mathbb{R}^{n_{\ell+1}, n_\ell}$*, also called connection matrices*
- *biases* $b^{(\ell)} \in \mathbb{R}^{n_{\ell+1}}$
- *both iid standard normals* $\mathcal{N}(0,1)$ *element-wise*

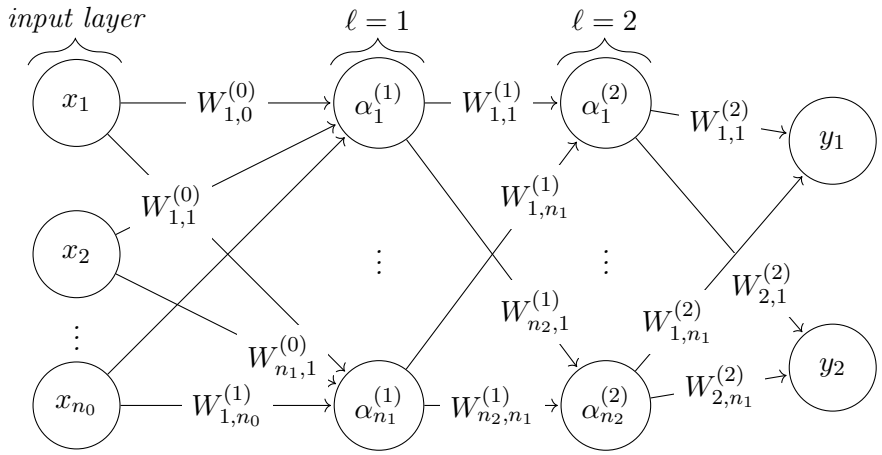


Figure 1: Notice that the bias is missing for the sake of making the image clear, we present a simpler but graphically correct form in Figure 3.

A common choice of initialization is the so-called LeCun initialization [LeC+12], corresponding to $W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_\ell}\right)$ *and* $b_j^{(\ell)} \sim \mathcal{N}(0,1)$*, or even* $b_j^{(\ell)} = 0$*.*

When considered jointly, we use the symbol $\theta = \left\{\left(W^{(\ell)}, b^{(\ell)}\right)\right\}_{\ell=1}^{L-1}$*. For simplicity, we will express* $\theta \in \mathbb{R}^P$ *where* $P = \sum_{\ell=1}^{L-1}(n_\ell + 1)n_{\ell+1}$*.*

We aim to estimate a function of the form:

$$f_\theta(x) = W^{(L-1)}\left(\sigma\left(W^{(L-2)}\left(\sigma\left(\cdots\sigma\left(W^{(0)}x + b^{(0)}\right)\right)\right) + b^{(L-2)}\right)\right) + b^{(L-1)} \tag{I.2.3}$$

The objective is to efficiently approximate $\vec{\boldsymbol{y}}$ *according to a parametric loss* $\mathscr{L} : \mathbb{R}^P \to \mathbb{R}_+$*. Having formulated the problem in terms of* θ*, we search:*

$$\theta^* = \arg\min_{\theta\in\mathbb{R}^P} \mathscr{L}\left(\theta; \vec{\boldsymbol{y}}, \boldsymbol{X}\right) = \arg\min_{\theta\in\mathbb{R}^P} \sum_{i=1}^{N} \mathcal{L}(\theta, y_i, x_i)$$

Where the last two arguments are fixed for both functions $\mathscr{L}, \mathcal{L}$*. Notice that* $\mathscr{L}$ *is just the sum over the dataset of* $\mathcal{L}$*.*

In the process of training we would not consider θ *as a parameter but optimize over it.*

Example I.2.4 (A $L = 3$ Neural Network)**.** *The diagram in Figure 1 shows an intuitive representation of a Neural Network architecture as per the description of Definition I.2.2. The inputs are* n_0 *dimensional. They are passed to activations* $\alpha^{(1)}$ *of the form* $\alpha_i^{(1)}(x) = \sigma\left(W_{i,\bullet}^{(1)}x + \beta b^{(1)}\right) = \sigma\left(\widetilde{\alpha}^{(1)}(x)\right)$*. These are in turn passed to the second hidden layer and eventually the output, with size* $n_L = 2$ *for* $L = 3$*. In the representation,* ***we are ignoring the added bias****, which for a layer* ℓ *is a neuron that adds* $\beta b^{(\ell)}$ *to the neurons of the subsequent layer. This is avoided to make the picture (hopefully clearer).*

Observation I.2.5 (On the architecture)**.** *The construction above is already satisfactory to establish a well defined learning problem, solvable with gradient descent and variants of it.*

Assumption I.2.6 (Input distribution form)**.** *Throughout the document,* p^{in} *is the empirical measure on the finite dataset, namely:*

$$p^{in} = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i} \quad x_i \in \mathcal{D}\ \forall i$$

Assumption I.2.7 (Nonlinearities form)**.** *The nonlinearities $\sigma : \mathbb{R} \to \mathbb{R}$ are:*

- *twice differentiable*
- *Lipschitz (Def. A.2.1)*
- *with bounded second derivative*

We provide a slightly varied perspective on the problem, which will be used as a starting point for the main results of the document.

Definition I.2.8 (ANN, functional view)**.** *Let:*

- *the parameters be initialized element-wise as standard Gaussians $\mathcal{N}(0,1)$*
- *Assumption I.2.6 hold*
- *Assumption I.2.7 hold*

For an ANN, consider the ***realization function****:*

$$F^{(L)} : \mathbb{R}^P \to \mathcal{F} \quad \theta \to f_\theta(x)$$

Where $\mathcal{F}$ is the set of functions that can be represented by the parameters as a ***network function*** *$f_\theta(x)$ in a form* ***similar to*** *that of Equation I.2.3 (see below).*
The ANN ***function space*** *is*

$$\mathcal{F} = \{f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}\}$$

Namely, the space of functions that start from the input space $\mathscr{X}$ and map to the output space $\mathscr{Y}$. We couple it with the seminorm (Def. A.1.12) induced by a fixed p^{in}, which has a natural notion of inner product (a positive definite symmetric bilinear form, Defs. A.1.5, A.1.7):

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

where the induced seminorm is explicitly:

$$\|\cdot\|_{p^{in}} : \mathcal{F} \to \mathbb{R}_+ \quad \|f\|_{p^{in}} = \mathbb{E}_{x \sim p^{in}} \left[f(x)^T f(x) \right]$$

The authors implement a slighlty tweaked form of the network function, which can be expressed in recursive form by a set of preactivations $\widetilde{\alpha}^{(\ell)}$ and activations $\alpha^{(\ell)}$ which for $\ell \in 0, \dots, L$ are of the form:

$$\widetilde{\alpha}^{(\ell)} : \mathscr{X} \to \mathbb{R}^{n_\ell} \quad \alpha^{(\ell)} : \mathscr{X} \to \mathbb{R}^{n_\ell} \quad \mathscr{X} \subseteq \mathbb{R}^{n_0}$$

and are identified by the recursive relation:

$$\begin{aligned} \alpha^{(0)}(x;\theta) &= x \\ \widetilde{\alpha}^{(\ell+1)}(x;\theta) &= \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \alpha^{(\ell)}(x;\theta) + \beta b^{(\ell)} \quad \beta > 0 \\ \alpha^{(\ell)}(x;\theta) &= \sigma\left(\alpha^{(\ell)}(x;\theta)\right) \end{aligned} \tag{I.2.9}$$

We set $f_\theta(x) = \widetilde{\alpha}^{(L)}(x;\theta)$, notice that we specifically use the preactivation to have a final linear combination. This is just a concise form to express the output of a Neural Network, with the advantage of having a way to summon the various stages of learning across layers, which are

precisely $\widetilde{\alpha}^{(\ell)}, \alpha^{(\ell)}$.
When optimizing, we do so with respect to a ***functional cost****:*

$$C : \mathcal{F} \to \mathbb{R}$$

which can be:

- *regression (see Subsection IV.2 for a final application)*
- *cross entropy*

and feed our realization function $F^{(L)}$ *to it, namely:*

$$\theta^* = \arg\min_{\mathbb{R}^P} \left\{ (C \circ F^{(L)})(\theta) \right\} \quad \mathscr{L} = C \circ F^{(L)} : \mathbb{R}^P \to \mathbb{R}$$

where the loss is $\mathscr{L} = C \circ F^{(L)}$.

Observation I.2.10 (Comments on the Realization function construction)**.** *There are clear differences between the canonical construction of Definition I.2.2 and Definition I.2.8, which is the one chosen in the publication.*

- *initializations of the parameters are different*
- $\beta > 0$*, the red term is added*
- *a* $\frac{1}{\sqrt{n_\ell}}$ *factor is added in front of each preactivation*

The reasons for doing this are that they serve as a form of ***scaling*** *to observe the asymptotic regime of diverging number of neurons at each layer* $n_1, , \ldots, n_{L-1}$*. We stress that:*

- *the representable space* $F^{(L)}(\mathbb{R}^P)$ *is the same*[1]
- *the formulations of the derivatives* $\partial_{W^\ell_{ij}} F^{(L)}, \partial_{b^\ell_j} F^{(L)}$ *are scaled by a factor of* $\frac{1}{\sqrt{n_\ell}}, \beta$ *respectively*

This consistent asymptotics regime comes at the price that we highly constraint the potential of the connection matrices (indeed, we will let $n_\ell \to \infty$*), so the* β *can be seen as a balance term that could outweigh this forced lower expressiveness of the layer.*

Observation I.2.11 (On the non-linearity assumption)**.** *The form of the non-linearities, other than being useful for simplifying proofs, does not seem to be strictly required [JGH20].*

Observation I.2.12 (Cost optimization hardness)**.** *A common and reasonable choice is assuming a convex functional cost* C*. Even in this case, the composite function* $C \circ F^{(L)}$ *is known to be highly non-convex [Cho+15]. This is the main reason why backpropagation does not converge to global minima.*

In this setting, the idea of the authors is to analyze the dynamics of such model at a functional level where convexity is guaranteed, focusing on f_θ. It will be shown that gradient descent induces an explicit trajectory along the kernel gradient, renamed as **Neural Tangent Kernel** (NTK).

Example I.2.13 (Linear Regression & Basics)**.** *A very well known model is regression. In its simplest formulation we have a dataset* $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ *where* $x_i \in \mathscr{X} \subset \mathbb{R}^{n_0}$ *and* $y_i \in \mathscr{Y} \subset \mathbb{R}$*. We wish to fit a function:*

$$f(W, x, b) = W^T x + b$$

[1] intuitively a scaling factor can be made up by learnable parameters in θ

which with the formalism of Definition I.2.2 loosely means a $L = 1$ layer ANN with scalar output dimension $n_L = 1$ and $\theta = \{(W, b)\}$. By loosely, it is intended that linear regression is thought to be an independent problem in most cases, and not a special case of Neural Networks. It is also agreed that its formulation is such that the two are basically different under every aspect.
For simplicity we denote the estimator with:

$$\widehat{y} = f_\theta(x) = \theta^T x$$

by letting b inside W and attaching a 1 to the vector x, so $\theta \in \mathbb{R}^{n_0+1}$. Training is formalized as a minimization of a loss function $\mathscr{L}(\theta)$ over which we choose suitable weights that minimize it. Namely, we aim to find:

$$\arg \min_{\theta \in \mathbb{R}^{n_0+1}} \mathscr{L}(\theta)$$

A basic choice is the square loss:

$$\mathscr{L}(\theta) = \frac{1}{2} \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2$$

It turns out that an easy optimization schedule is Gradient Descent (GD), which for a time t can be defined by the equations:

$$\begin{aligned} \theta(t+1) &= \theta(t) - \eta_{(t)} \partial_\theta \mathscr{L}(\theta(t)) \\ \partial_\theta \mathscr{L}(\theta) &= \sum_i y_i - f_\theta(x_i) \partial_\theta f_\theta(x_i) \\ &= \sum_i y_i - f_\theta(x_i) \boldsymbol{x_i} \end{aligned}$$

where $\eta_{(t)}$ is the learning rate, potentially time dependent. Notice that $\partial_\theta f$ is static and independent of time t and $\theta(t)$. This is a consequence of linearity of the model.

For a sufficiently small learning rate $\eta^{(t)}$ and advantageous conditions it is possible to converge to a global optimizer. One necessary requirement is convexity of $\mathscr{L}$. However, this is clearly non satisfactory as we optimize over the training space and do not have a clear representation of the whole distribution of the joint of $\mathscr{X}$ and $\mathscr{Y}$. For this reason, one would need a suited discussion on the generalization properties that we avoid.
Non-linear models and variations of linear regression are more customizable, and thus achieve better performance on hard tasks. However, the trade-off of higher flexibility is less guarantees on explainability and performance.

I.3 Kernels for Functionals

Despite being a simple idea, the theoretical results are not immediate. For those interested, a starting point could be [SC04] which is completely dedicated to the topic, and [HTF09](Chapters 5, 12) for the Machine Learning perspective. Additionally, an interesting survey paper on Kernels for Machine Learning is [Gho+21].

We consider a transformation of the inputs x_i to a higher dimensional space via a feature map:

$$x_i \in \mathbb{R}^{n_0} \rightsquigarrow \varphi(x_i) \in \mathbb{R}^D \quad D \gg n_0$$

Example I.3.1 (Easy feature map)**.** *Let* $n_0 = 3$ *and imagine the feature map is a function* $\varphi : \mathbb{R}^3 \to \mathbb{R}^5$ *such that:*

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightsquigarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1 x_2 \\ x_2 x_3 \end{bmatrix}$$

After such transformation, if we still used linear regression as in Example I.2.13, we would be minimizing:

$$\min_\theta \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T \varphi(x_i))^2 \quad \theta \in \mathbb{R}^{D+1}$$

Is this model linear in θ? Yes, we are just transforming data, not parameters. This is not akin to Neural Networks, which are non-linear in θ. On the other hand, it is not linear in x. Regardless, optimization over θ is not affected, apart from requiring higher computational resources. **Why is it good**? The main advantage is that in a higher dimensional space the problem might be easier to solve, in the sense that a linear problem could solve it, while it might have not with a lower dimensional linear regression construction as in Example I.2.13. We take this as granted, but suggest to give a look at Cover's Theorem [Cov65]. A proof of it is in [MMR97](Section 3.5), or in an intuitive form at this link.
Why is this not satisfactory? Notice that:

- φ is fixed
- we might suffer from one of the problems associated with the curse of dimensionality, explained below with an Example.

this list is not exhaustive.

Example I.3.2 (Curse of Dimensionality)**.** *Consider a polynomial* $\varphi(x) \in \mathbb{R}^D$ *where* $D \gg n_0$ *and degree* $\leq d$*. It holds that* $D = O(n_0^d)$*. For ImageNet [Den+09], where* $n_0 \sim 10^5$*, if* $d = 3$ *then* $D \sim O(10^{15})$*. Optimizing over this space is hard.*

Here comes the kernel trick. Imagine we have a function that is a notion of similarity. This could be loosely defined as an inner product over a higher dimensional space, or equivalently by Mercer's Theorem[2] a positive semi-definite Kernel in the sense of Definition A.2.4. Then:

$$\mathbf{K} : \mathscr{X} \times \mathscr{X} \to \mathbb{R}_+ \quad \mathbf{K}(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$$

Notice that the inputs are in the dimension of the dataset $\mathcal{D}$, but the output is effectively an inner product in a D-dimensional space. Additionally, the kernel function forms a **positive semi-definite** (Def. A.2.4) $N \times N$ Gram Matrix $\widetilde{\mathbf{K}}$ (Def. A.2.12).
The higher dimensional transformation might not be required if we are able to represent distances in this space via its inner product. This implicit computation saves computational resources while still providing the information of the expensive computation. To do so, it is sufficient to possess a closed form expression of the inner product. If the optimization technique depends only on the kernel values, then the performance is as good as transforming explicitly all datapoints.

[2] we briefly touch upon it and provide further references in Definition A.2.4

Example I.3.3 (Polynomial kernels)**.** *consider:*

$$\boldsymbol{K}(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle = (c + x_i^T x_j)^d \quad \varphi(x_i) \in \mathbb{R}^D \implies D = O(n_0^d)$$

but the computation is $O(n_0)$, since we know the binomial expansion of coefficients and just need to multiply the powers. The operation is performed on the lower dimensional space while representing the higher dimensional space complexity.

We find implementations of this concept in many successful methods such as:

- Kernel SVM and Kernel Ridge regression [SC04]
- Kernelized PCA[SSM97]

Such methodology justifies expanding its formulation to be applicable to the learning setting of Definition I.2.2.

Definition I.3.4 (Kernel, multidimensional version)**.** *A function:*

$$\boldsymbol{K} : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L} \quad (x, x') \to \boldsymbol{K}(x, x')$$

Where $\boldsymbol{K}(\cdot, \cdot)$ is a matrix, with the property that:

$$\boldsymbol{K}(x, x') = \boldsymbol{K}(x', x)^T \quad \forall x, x'$$

The whole requirements are equivalently expressed by stating that $\boldsymbol{K}$ is a symmetric tensor in $\mathcal{F} \otimes \mathcal{F}$ (Def. A.1.13, Ex. A.1.14).
This definition is a non ambiguous multidimensional generalization of Definition A.2.4, which maps to the real line only.

It is possible to build a bilinear form (Def. A.1.5) tweaked by the Kernel representation in the space of functions in the sense that:

$$\langle f, g \rangle_{\mathbf{K}} \coloneqq \mathbb{E}_{x,x' \sim p^{in}} \left[f(x)^T \mathbf{K}(x, x') g(x') \right] \quad \forall f, g \in \mathcal{F}, \quad x \perp\!\!\!\perp x' \tag{I.3.5}$$

Definition I.3.6 (Positive definiteness of Kernel wrt a norm)**.** *We say a Kernel is positive definite (p.d.) with respect to the distribution norm $\|\cdot\|_{p^{in}}$ if the following condition holds:*

$$\forall f : \|f\|_{p^{in}} > 0 \implies \|f\|_{\boldsymbol{K}} = \langle f, f \rangle_{\boldsymbol{K}} = \mathbb{E}_{x,x' \sim p^{in}} \left[f(x)^T \boldsymbol{K}(x, x') f(x') \right] > 0$$

Which basically means that at non null in norm functions wrt to the empirical distribution the kernel matrix is positive definite in expectation.

Observation I.3.7 (Comments on p.d. wrt empirical norm)**.** *We compare this Definition with the classical ones (Defs. A.1.7, A.2.4). In this case it might as well be that for a function that has zero $\|\cdot\|_{p^{in}}$ norm the kernel might be non positive definite at the evaluation. However, it must be noted that this is just a matter of choice of the reference norm.*

I.3.1 Dual notions

The dual (Def. A.1.10) of the space of functions $\mathcal{F}$ wrt p^{in} is denoted as $\mathcal{F}^*$, and in this case is the set of linear functionals:

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

Namely, we have $\mathcal{F}^* = \left\{ \mu = \langle d, \cdot \rangle_{p^{in}} \ \ for\ some\ d \in \mathcal{F} \right\}$.

Observation I.3.8 (On the Dual space)**.** *Recognize that*

- *the choice $d \in \mathcal{F}$ is arbitrary*
- *being that we are restricting ourselves to the empirical distribution, it holds that two functions f, g have same linear form if they are equal on the data:*

$$f, g \in \mathcal{F} : f \sim g \iff \mu = \langle f, \cdot \rangle_{p^{in}} = \langle g, \cdot \rangle_{p^{in}} = \nu$$

which is an equivalence relation. Again the expectation is over the empirical distribution. If we consider the quotient space:

$$\left(\mathcal{F} \setminus \{f, g : f \sim g\}, \langle \cdot, \cdot \rangle_{p^{in}} \right) \tag{I.3.9}$$

then we have a finite dimensional space. Thus, it is also a Hilbert space, to which the dual space $\mathcal{F}^*$ is isomorphic more details are given by Mateusz Mroczka and Benedikt Petko(Section 1.3).

For a kernel $\mathbf{K}$, consider its partial application, where in this case *loosely* means:

- fix the row
- fix the left input
- let the column index and the right input vary

and precisely[3]:

$$\mathbf{K}_{i,\bullet}(x, \bullet) = \begin{bmatrix} K_{i,1}(x, \cdot) \\ \vdots \\ K_{i,n_L}(x, \cdot) \end{bmatrix} \in \{f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}\} = \mathcal{F} \quad i \in \{1, \dots, n_L\}, x \in \mathbb{R}^{n_0}$$

Where the i is arbitrary and is fixed just to extract a row. For a fact, in such row, we have a fixed $x \in \mathbb{R}^{n_0}$. Such a construction is an element of our function space and we could devise a map:

$$\Phi_{\mathbf{K}} : \mathcal{F}^* \to \mathcal{F} \quad \mu = \langle d, \cdot \rangle_{p^{in}} \to \Phi_{\mathbf{K}}(\mu) = f_\mu \in \mathcal{F}$$

where the function f_μ, indexed by its **dualized** parameter has form:

$$f_{\mu,i}(x) = \mu \left(\mathbf{K}_{i,\bullet}(x, \bullet) \right) = \langle d, \mathbf{K}_{i,\bullet}(x, \bullet) \rangle_{p^{in}} \quad i \in \{1, \dots, n_L\}, x \in \mathbb{R}^{n_0} \tag{I.3.10}$$

Notice that already x has in principle no restriction and is free to vary. This is the basis of the idea we outline below.

In the context of a sample from a population such as that of Definition I.2.1, there are N individuals. By the way in which the cost C is designed, it can only access evaluations of candidate functions $f \in \mathcal{F}$ at the (hyper)points $x_1, \dots, x_n \in \mathbb{R}^{n_0}$. The functional derivative (Def. A.2.3) is constructed with respect to the inner product $\langle \cdot, \cdot \rangle_{p^{in}}$ and takes form:

$$\frac{\delta C}{\delta f} \in \mathcal{F} : \quad \left\langle \frac{\delta C}{\delta f}, \phi \right\rangle_{p^{in}} = \lim_{\epsilon \downarrow 0} \frac{C(f + \epsilon\phi) - C(f)}{\epsilon}$$

[3]Here we use the bullet `\bullet` notation for variable inputs to highlight it at the pedix, consider it to be the canonical `\cdot` $\cdot$

where for simplicity we denote $\frac{\delta C}{\delta f} \coloneqq d|_f$. Then, the representative dual is:

$$\partial_f C^{in} : \mathcal{F} \to \mathcal{F}^* \quad f_0 \to \partial_f C^{in}|_{f_0} = \langle d|_{f_0}, \cdot \rangle_{p^{in}} \in \mathcal{F}^* \quad \forall f_0 \in \mathcal{F}$$

Notice that we also make specific the dual element of f_0, which is $d|_{f_0}$. The difference with $\mu = \langle d, \cdot \rangle_{p^{in}}$ is that by evaluating the functional derivative at a specific $f_0 \in \mathcal{F}$, we induce a specific linear functional $\partial_f^{in} C|_{f_0}$. The identification is well explained in Mateusz Mroczka and Benedikt Petko(Sec. 1.4), and is based on a thoughtful application of the Riesz-Markov-Kakutani Theorem on the quotient space of Equation I.3.9.

Definition I.3.11 (Kernel Gradient)**.** *We define the Kernel gradient to be the function induced by the **dualized** parameter $\partial_f^{in} C|_{f_0}$ as in Equation I.3.10. Namely:*

$$\nabla_{\boldsymbol{K}} C|_{f_0} \coloneqq \Phi_{\boldsymbol{K}} \left(\partial_f^{in} C|_{f_0} \right) : \mathcal{F}^* \to \mathcal{F}$$

*Where its space is not only defined on the dataset like $\partial_f^{in} C|_{f_0}$. Indeed, by the kernel being **dataset agnostic**, we are effectively extending the space of the gradient to $\mathbb{R}^{n_0}$ completely. Our new function $\nabla_{\boldsymbol{K}} C|_{f_0}$ will take an input element $x \in \mathbb{R}^{n_0}$ and return an output in the space $\mathbb{R}^{n_L}$ as:*

$$\nabla_{\boldsymbol{K}} C|_{f_0} = \frac{1}{N} \sum_{j=1}^{N} K(x, x_j) d|_{f_0}(x_j) \tag{I.3.12}$$

which is a notion of similarity of the dataset information, here indexed by j, and the input x.

An arbitrary time-indexed function $f(t)$ follows a **kernel gradient descent**, which in this case is a **gradient flow** if its dynamics are described by the differential equation:

$$\partial_t f(t) = -\nabla_{\mathbf{K}} C|_{f(t)} = -\Phi_{\mathbf{K}} \left(\partial_f^{in} C|_{f(t)} \right) = -\Phi_{\mathbf{K}} \left(\langle d|_{f(t)}, \cdot \rangle_{p^{in}} \right)$$

A nice introduction to gradient flows via gradient descent is given in [Bac20]. There are also guarantees that it is not incorrect to use it if the objective is gradient descent or stochastic gradient descent. We take this notion as granted, a longer discussion is in a document I recently prepared at this link. For a visualization of how the dynamics differ, Figure 2.

Imposing this update of the function, the cost will evolve as:

$$\begin{aligned}
\partial_t C|_{f(t)} &= \partial_f^{in} C|_{f(t)} \partial_t f(t) && \text{chain rule} \\
&= \langle d|_{f(t)}, \partial_t f(t) \rangle \\
&= - \langle d|_{f(t)}, \nabla_{\mathbf{K}} C|_{f(t)} \rangle_{p^{in}} \\
&= -\mathbb{E}_{x \sim p^{in}} \left[\left(d|_{f(t)}(x) \right)^T \left(\nabla_{\mathbf{K}} C|_{f(t)}(x) \right) \right] \\
&= -\mathbb{E}_{x \sim p^{in}} \left[\left(d|_{f(t)}(x) \right)^T \left(\frac{1}{N} \sum_{j=1}^{N} K(x, x_j) d|_{f(t)}(x_j) \right) \right] \\
&= -\mathbb{E}_{x, x' \sim p^{in}} \left[\left(d|_{f(t)}(x) \right)^T \mathbf{K}(x, x') \left(d|_{f(t)}(x') \right) \right] \\
&= - \left\| d|_{f(t)} \right\|_{\mathbf{K}}
\end{aligned}$$

Observation I.3.13 (Kernel sufficient conditions for convergence)**.** *For the cost to be converging to a critical point the kernel needs to be positive definite (Def. I.3.6). This would ensure that the evolution follows a direction along which the cost decreases until $f(t)$ is a stationary point.*

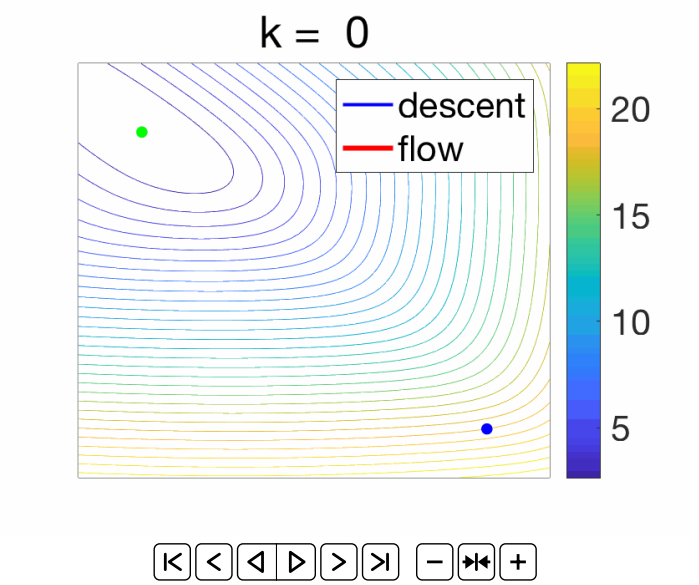


Figure 2: Descent vs. Flow. Source [Bac20]
Gradient descent vs. gradient flow on the same time scale for a logistic regression problem. Time is indexed by k.

By the peculiarity of having a differential equation in the space of functions $\mathcal{F}$, cost **convexity and boundedness from below** imply that the dynamics converge to a global minima as time diverges, $t \to \infty$. This is intuitively verified since the derivative over time of the cost is negative, convexity ensures the uniqueness of the global minima and the boundedness from below implies that there are no asymptotes.
Despite this interesting result, we will see that more details are needed when considering non-linear models such as Neural Networks.

I.4 Random functions approximation

Consider a kernel $\mathbf{K}$. It can be approximated [JGH20] by a choice of $P = \sum_{\ell=0}^{L-1}(n_\ell + 1)n_{\ell+1}$ random functions $\{f^{(p)}\}_{p=1}^{P} \subset \mathcal{F}$ independently sampled with covariance given by the kernel itself:

$$\mathbb{E}\left[f_k^{(p)}(x) f_{k'}^{(p)}(x')\right] = \mathbf{K}_{k,k'}(x, x')$$

A linear combination of these functions is a random liner parametrization of the form:

$$F^{lin} : \mathbb{R}^P \to \mathcal{F} \quad \theta \to f_\theta^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \theta_p f^{(p)}$$

We loosely interpret this as sampling random functions from a distribution and finding the best linear combination that fits a cost.
The model F^{lin}, linear in the parameters θ_p, has partial derivative:

$$\partial_{\theta_p} F^{lin}(\theta) = \frac{1}{\sqrt{P}} f^{(p)} \quad \forall \theta \in \mathbb{R}^p$$

Using gradient descent on the composite cost $C \circ F^{lin}$, we will work at a parameter level. Using the notation $f_{\theta(t)}^{lin} := F^{lin}(\theta(t))$ establish that:

- the parameters are governed by

$$\begin{aligned}\partial_t \theta_p(t) &= -\partial_{\theta_p} \mathscr{L}(\theta(t)) \\ &= -\partial_{\theta_p} \left(C \circ F^{lin} \right)(\theta(t)) \\ &= -\partial_f^{in} C|_{f^{lin}(\theta(t))} \partial_{\theta_p} F^{lin}(\theta) \\ &= \frac{1}{\sqrt{P}} \partial_f^{in} C \Big|_{f^{lin}_{\theta(t)}} f^{(p)} \\ &= -\frac{1}{\sqrt{P}} \left\langle d|_{f^{lin}_{\theta(t)}}, f^{(p)} \right\rangle_{p^{in}}\end{aligned}$$

- the network function $f^{lin}_{\theta(t)}$ derivative is easily found by plugging the above result as:

$$\begin{aligned}\partial_t f^{lin}_{\theta(t)} &= \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \partial_t (\theta_p(t) f^{(p)}) \\ &= \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \partial_t (\theta_p(t)) f^{(p)} && f^{(p)} \text{ is constant in } t \\ &= \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \left(-\frac{1}{\sqrt{P}} \left\langle d|_{f^{lin}_{\theta(t)}}, f^{(p)} \right\rangle_{p^{in}} \right) f^{(p)} && \text{above result} \\ &= -\frac{1}{P} \sum_{p=1}^{P} \left\langle d|_{f^{lin}_{\theta(t)}}, f^{(p)} \right\rangle_{p^{in}} f^{(p)} \\ &= -\Phi_{\widetilde{\boldsymbol{K}}} \left(\partial_f^{in} C|_{f^{lin}_{\theta(t)}} \right) \\ &= -\nabla_{\widetilde{\boldsymbol{K}}} C|_{f^{lin}_{\theta(t)}}\end{aligned}$$

Observation I.4.1 (The update function is a kernel gradient)**.** *The last expression of the function update over time is equivalent to the Kernel gradient (Def. I.3.11) $-\nabla_{\widetilde{\boldsymbol{K}}} C$ where:*

$$\widetilde{\boldsymbol{K}} = \sum_{p=1}^{P} \partial_{\theta_p} F^{lin}(\theta) \otimes \partial_{\theta_p} F^{lin}(\theta) = \frac{1}{P} \sum_{p=1}^{P} f^{(p)} \otimes f^{(p)} \tag{I.4.2}$$

which is random, $n_L \times n_L$ dimensional, and has entries:

$$\widetilde{\boldsymbol{K}}_{ii'}(x, x') = \frac{1}{P} \sum_{p=1}^{P} f_i^{(p)}(x) f_{i'}^{(p)}(x')$$

We just established that gradient descent on the parametric function $C \circ F^{lin}$ is equivalent to functional kernel descent on a random tangent kernel acting on interpolating functions. As parameters diverge, by the Law of Large Numbers, $\widetilde{\mathbf{K}} \to \mathbf{K}$ the limiting Kernel. At finite but big enough P, we can say that $\widetilde{\mathbf{K}}$ is its approximation, or simply use $\mathbf{K}$ as a surrogate of the finite dimensional dynamics. The latter option will be studied for reasons that will be clear once needed.

In a nutshell Using the result that ANNs as in Definition I.2.2 are Gaussian processes if all hidden layers diverge [Nea96; DFS17; Mat17; Lee+18; Mat+18], we will build a description of them via kernel methods. Explicitly, the network function will be shown to obey a Neural Tangent Kernel gradient with respect to the functional cost. Such result, though random at initialization and variable during training by the non linearity of ANNs, is, under precise assumptions, constant. Convergence to an optimal configuration will also be related to the positive definiteness of the NTK, with a nice motivation for early stopping heuristics. Such construction is proved to be well-suited in many scenarios. Lastly, experimental results show to be favorable against this hypothesis, with some limitations.

II Neural Tangent Kernel

After having cleared the structure of the problem, we dedicate some space to a hopefully thorough explanation of the object of study, which is somehow complicated in its form. For this reason, to stimulate comprehension, a simple Example is reported in the second part.

II.1 General Form

Consider an Artificial Neural Network with the form of Definition I.2.2. If we perform gradient descent on the functional cost as proposed in Subsection I.3 the result is similar as Subsection I.4. Infact, imposing as differential equation the classic gradient flow a Kernel Gradient (Def. I.3.11) arises. The calculations are similar to Subsection I.4.

Lemma II.1.1 (NTK derivation)**.** *Consider an ANN as in Definition I.2.2, with the functional view of Definition I.2.8. Imposing gradient descent on the parameters:*

$$\partial_t \theta = -\partial_\theta \mathscr{L}(\theta(t)) \quad \mathscr{L} = C \circ F^{(L)}$$

the network function obeys a differential equation with respect to a kernel gradient:

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

where

$$\boldsymbol{\Theta}^{(\boldsymbol{L})}(\theta) = \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta)$$

Proof. **(parameter level)** we have a P dimensional vector where each entry evolves as:

$$\partial_t \theta_p = \partial_f^{in} C|_{f(\theta)} \partial_{\theta_p} f_\theta = -\left\langle d|_{f_\theta}, \partial_{\theta_p} f_\theta \right\rangle_{p^{in}}$$

using the chain rule.

(↰ realization function level) it holds:

$$\begin{aligned}
\partial_t f_{\theta(t)}(x) &= \sum_{p=1}^{P} \partial_{\theta_p} f_{\theta(t)}(x) \partial_t \theta_p(t) \\
&= -\sum_{p=1}^{P} \left\langle d|_{f_{\theta(t)}}, \partial_{\theta_p} f_\theta \right\rangle_{p^{in}} \partial_{\theta_p} f_{\theta(t)}(x) \\
&= -\frac{1}{N} \sum_{p=1}^{P} \sum_{i=1}^{N} \partial_{\theta_p} f_{\theta(t)}(x) \left(\partial_{\theta_p} f_{\theta(t)}(x_i)\right)^T d|_{f_\theta}(x_i) \\
&= -\Phi_{\boldsymbol{\Theta}^{(L)}} \left(\partial_f^{in} C|_{f_{\theta(t)}}\right) \\
&= -\nabla_{\boldsymbol{\Theta}^{(L)}} C|_{f_{\theta(t)}}
\end{aligned}$$

□

Definition II.1.2 (Neural Tangent Kernel, NTK)**.** *When the dynamics are* $\partial_t f_{\theta(t)} = -\nabla_{\boldsymbol{\Theta}^{(L)}} C\Big|_{f_{\theta(t)}}$ *we say that the NTK is:*

$$\mathbb{R}^{n_L \times n_L} \ni \boldsymbol{\Theta}^{(L)}(\theta) = \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta) \tag{II.1.3}$$

For elements of $\mathscr{X}$ *an entry has form:*

$$\boldsymbol{\Theta}_{ij}^{(L)}(\theta)(x, x') = \sum_{p=1}^{P} \left[\partial_{\theta_p} F^{(L)}(\theta, x)\right]_i \left[\partial_{\theta_p} F^{(L)}(\theta, y)\right]_j \quad x, x' \in \mathscr{X} \tag{II.1.4}$$

while as a matrix the explicit representation is:

$$\begin{aligned}
\boldsymbol{\Theta}^{(L)}(\theta)(x, x') &= \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta, x) \otimes \partial_{\theta_p} F^{(L)}(\theta, x') \\
&= \sum_{p=1}^{P} \begin{bmatrix} \partial_{\theta_p} F_1^{(L)}(\theta, x) \\ \vdots \\ \partial_{\theta_p} F_{n_L}^{(L)}(\theta, x) \end{bmatrix} \otimes \begin{bmatrix} \partial_{\theta_p} F_1^{(L)}(\theta, x') \\ \vdots \\ \partial_{\theta_p} F_{n_L}^{(L)}(\theta, x') \end{bmatrix} \\
&= \sum_{p=1}^{P} \begin{bmatrix} \partial_{\theta_p} F_1^{(L)}(\theta, x) \partial_{\theta_p} F_1^{(L)}(\theta, x') & \cdots & \partial_{\theta_p} F_1^{(L)}(\theta, x) \partial_{\theta_p} F_{n_L}^{(L)}(\theta, x') \\ \vdots & \ddots & \vdots \\ \partial_{\theta_p} F_{n_L}^{(L)}(\theta, x) \partial_{\theta_p} F_1^{(L)}(\theta, x') & \cdots & \partial_{\theta_p} F_{n_L}^{(L)}(\theta, x) \partial_{\theta_p} F_{n_L}^{(L)}(\theta, x') \end{bmatrix}
\end{aligned}$$

Observation II.1.5 (About the Kernel)**.** *Notice that, differently from Equation I.4.2 we are specifically expressing the dependence on* θ*. This is due to the fact that* $F^{(L)}$ *is non linear in* θ*, and* ∂_{θ_p} *is not independent of* θ*.*
For this reason, we could say that the NTK is random at initialization (by the random weights initialization), and changes with the iterations. Again, differently from our dummy example, it is worth stressing that this does not mean that parameters in the random functions approximation do not change, but rather that the underlying kernel does not.

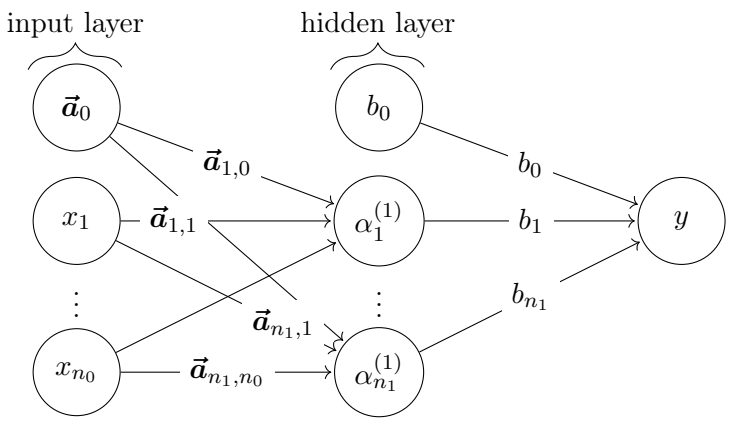


Figure 3: The diagram shows an intuitive representation of a two-layer neural network. The inputs are n_0 dimensional, with an added bias. They are passed to activations $\alpha_i^{(1)}$ of the form $\alpha_i^{(1)}(x) = \sigma(\widetilde{\alpha}_i^{(1)})$. The final output is then determined by a weighted sum of activations. With the architecture considered, $,\vec{\boldsymbol{a}}_0 = \beta\vec{\mathbf{1}}, \beta_0 = \beta$ and $\widetilde{\alpha}^{(1)}, \widetilde{\alpha}^{(2)}$ have the scaling factors $\frac{1}{\sqrt{n_\ell}}$ inside.

II.2 One Layer Neural Networks

We provide an intuitive Example for Definition II.1.2, overlooking some details.
Consider an ANN with one hidden layer and a scalar output. A graphical representation is given in Figure 3. Since we just want to show how a Kernel operates in general with these models we implement a simpler notation than that of Definition I.2.2. We have that:

- in the first layer, the parameters are $\{\vec{\boldsymbol{a}}_j\}_{j=1}^{n_1}$ for each neuron, with $\vec{\boldsymbol{a}}_0$ being the added bias. If we mean the i^{th} entry of the j^{th} vector, we will write a_{ji}. If we mean the j^{th} vector, we will use $\vec{\boldsymbol{a}}_j$
- in the second layer parameters are $\{b_j\}_{j=1}^{n_1}$ for each neuron, with b_0 being the added bias

the output can be written as:

$$\widehat{y}_i = f_\theta(x_i) = \sum_{j=1}^{n_1} b_j \sigma(\vec{\boldsymbol{a}}_j^T x_i)$$

Where σ is a non-linearity applied element-wise. With square loss minimization, it is again possible to perform gradient descent:

$$\theta(t+1) = \theta(t) - \eta_{(t)} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)\partial_\theta f_{\theta(t)}(x_i)$$

Using either $\phi(x)$ or x the gradient is not fixed and depends on how the parameters change over time. Notice that this is the gradient descent equation but we will work under the gradient flow regime regardless.
Going back to our NN setting, we express explicitly the dependence on n_1 in the function, with the rescaling of Definition I.2.2:

$$f_\theta(x) = \frac{1}{\sqrt{n_1}} \sum_{j=1}^{n_1} b_i \sigma(\vec{\boldsymbol{a}}_j^T x)$$

The gradient computation is

- $\partial_{\vec{\boldsymbol{a}}_j} f_\theta(x) = \frac{1}{\sqrt{n_1}} b_j \dot{\sigma}(\vec{\boldsymbol{a}}_j^T x)x$
- $\partial_{b_j} f_\theta(x) = \frac{1}{\sqrt{n_1}} \sigma(\vec{\boldsymbol{a}}_j^T x)$

Where all weights are normalized by the square root factor, and biases have a β factor instead of being unitary.
We concatenate their tensor products[4] to form the NTK kernel since the derivative operator is linear:

$$\mathbf{K}(x, x') = \mathbf{K}^{(a)}(x, x') + \mathbf{K}^{(b)}(x, x')$$

where precisely

- $\mathbf{K}^{(a)}(x, x') = \frac{1}{\sqrt{n_1}} \frac{1}{\sqrt{n_1}} \sum_{j=1}^{n_1} b_j^2 \dot{\sigma}(\vec{\boldsymbol{a}}_j^T x) \dot{\sigma}(\vec{\boldsymbol{a}}_j^T x')(x \cdot x')$
- $\mathbf{K}^{(b)}(x, x') = \frac{1}{n_1} \sum_{j=1}^{n_1} \sigma(\vec{\boldsymbol{a}}_j^T x) \sigma(\vec{\boldsymbol{a}}_j^T x')$

These values are easily computed. We have found the NTK of a specific instance of a one hidden layer ANN with width n_1. Informally, an independent initialization of $\vec{\boldsymbol{a}}_j, b_j$, ensures convergence to the expected value of the quantities by being a $\frac{1}{n_1}$ sum (i.e. an empirical average) **which justifies the rescaling**.
In other words, if $\{\vec{\boldsymbol{a}}_j\}, \{b_j\}$ are sampled independently, by the LLN as $n_1 \to \infty$ the former converges to:

$$\mathbf{K}^{(a)}(x, x') \overset{n_1 \to \infty}{\to} \mathbf{K}_{\infty}^{(a)}(x, x') = \mathbb{E}[b_j^2 \dot{\sigma}(\vec{\boldsymbol{a}}_j^T x) \dot{\sigma}(\vec{\boldsymbol{a}}_j^T x')(x \cdot x')]$$

with the other quantity being similar in principle. We will show this result formally in Section III, where the convergence conditions and the derivations are made clear.
Even though in practice an infinite-width limit is impossible to simulate, we will derive a closed-form solution which can be used as a linearized model. Such concept requires the results of Section III, and is presented in Subsection IV.1.

III Properties of the Neural Tangent Kernel

Having laid the foundations for a formal treatment, the four main results of [JGH20] are reported with proof. It is shown that network functions at initialization with divergent size of the hidden layers are equivalent to Gaussian Processes. This is instrumental to derive that the NTK is deterministic and independent of the parameters at initialization. For a more general class of training directions, uniform convergence across time is also true under Assumption I.2.7. Eventually, conditional on having spherical data, the NTK is positive definite and convergence to the global minima is theoretically proved.

III.1 Starting with sequential layer divergence

The learning setup is the architecture of Definition I.2.2. We now consider the behavior of the vector of functions:

$$f_\theta = \begin{bmatrix} f_{\theta,1} \\ \vdots \\ f_{\theta,n_L} \end{bmatrix} \in \mathcal{F}$$

[4] warning: in this case they are vectors

at the initialization. Clearly, the random start has some influence on the shape that the first hypothesized output function (i.e. no training) has. In particular, the focus is on the **infinite-width limit**, which means that all hidden layers diverge **sequentially**:

$$n_1 \to \infty, \dots, n_{L-1} \to \infty$$

Observation III.1.1 (On the sequential limit)**.** *Notice that:*

- *we are reasonably not increasing the input and output neurons since those are fixed at the dataset $\mathcal{D}$ dimensions*
- *we are specifying that the limits are taken sequentially and in order. This result is weaker than the general exchangeable limit:*

$$\lim_{n_1,\dots,n_{L-1}\to\infty} \quad i.e. \quad \lim_{\min\{n_1,\dots,n_{L-1}\}\to\infty}$$

Concerning the latter point the authors mentioned that more general results were in principle not impossible [JGH20]. A proof of this is shown in [Aro+19].
An important property already mentioned in [Nea96; DFS17; Lee+18; Mat+18; Dai+22] is proved below.

Proposition III.1.2 (Network functions are Gaussian processes at the limit)**.** *Let:*

- *the number of layers L be fixed*
- *the non-linearity σ be Lipschitz (Def. A.2.1)*

The limit:

$$\lim_{n_{L-1}\to\infty} \cdots \lim_{n_1\to\infty} f_{\theta,k} \quad k \in \{1, \dots, n_L\}$$

*is convergent **in law** to a collection of independent and identically distributed Gaussian processes (Def. A.2.6) with null mean and covariance defined recursively in L by the equations:*

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^T x' + \beta^2$$
$$\Sigma^{(L+1)}(x, x') = \mathbb{E}_{f\sim\mathcal{N}(0,\Lambda^{(L)})}\left[\sigma(f(x))\sigma(f(x'))\right] + \beta^2$$

Where the expectation notation means that we are sampling with respect to a centered Gaussian process f with covariance identified by the L layer size value, which is exactly:

$$\Lambda^{(L)}(x, x') = \begin{bmatrix} \Sigma^{(L)}(x, x) & \Sigma^{(L)}(x, x') \\ \Sigma^{(L)}(x', x) & \Sigma^{(L)}(x', x') \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

Remark. *The convergence of network functions is in terms of the k^{th} entry of the vector f_θ, so the entries **separately** converge. The covariance is across inputs, but not across dimensions $k = 1, \dots, n_L$ which are independent.*

Proof. (**strategy**) given the recursive claim, we prove it by induction on the depth of the network L.
(**base case** $L = 1$) no hidden layers are considered, inputs are linearly combined and

directly map to the output space. According to the architecture of Definition I.2.2 and the random initialization we will have that:

$$f_\theta(x) = \frac{1}{\sqrt{n_0}} W^{(0)} x + \beta b^{(0)}$$

which is both random and affine. Taking the outputs one by one with index $k = 1, \dots, n_L$ we notice that:

$$f_{\theta,k}(x) = \widetilde{\alpha}_k^{(1)}(x) = \frac{1}{\sqrt{n_0}} \sum_{i=1}^{n_0} W_{ki}^{(0)} x_i + \beta b_k^{(0)} \quad \forall x \in \mathbb{R}^{n_0}$$

The standard Gaussian initialization of the components of θ ensures that the mean is $\mu_{\theta_p} \equiv 0$ and the variance is identically one on each entry $\sigma^2_{\theta_p} \equiv 1$, with null covariance. This holds for any layer L provided that it is true for the first. Infact by a quick induction inside the induction:

$$\begin{aligned}
\mathbb{E}\left[\widetilde{\alpha}_k^{(L+1)}(x;\theta)\right] &= \mathbb{E}\left[\frac{1}{\sqrt{n_L}} W_{k,\bullet}^{(L)} \alpha^{(L)}(x;\theta) + \beta b_k^{(L)}\right] \\
&\quad \text{[independence]} \\
&= \frac{1}{\sqrt{n_L}} \mathbb{E}\left[W_{k,\bullet}^{(L)}\right] \mathbb{E}\left[\alpha^{(L)}(x;\theta)\right] + \mathbb{E}\left[\beta b_k^{(L)}\right] \\
&= 0 \\
\mathbb{E}\left[\widetilde{\alpha}_k^{(L+1)}(x;\theta)\widetilde{\alpha}_{k'}^{(L+1)}(x';\theta)\right] &= \mathbb{E}\left[\left(\frac{1}{\sqrt{n_L}} W_{k,\bullet}^{(L)} \alpha^{(L)}(x;\theta) + \beta b_k^{(L)}\right)\left(\frac{1}{\sqrt{n_L}} W_{k,\bullet}^{(L)} \alpha^{(L)}(x;\theta) + \beta b_k^{(L)}\right)\right] \\
&= 0
\end{aligned}$$

By similar arguments on independence and starting distribution which is $\mathcal{N}(0,1)$.
Eventually, the neurons $k \in \{1, \dots, n_1\}$ are iid. Given this construction, it is rather easy to recover the covariance at two datapoints:

$$\begin{aligned}
\Sigma^{(1)}(x,x') &= \mathbb{E}\left[\widetilde{\alpha}_k^{(1)}(x)\widetilde{\alpha}_k^{(1)}(x')\right] \\
&\quad \text{[by centering } \mathtt{CoV}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y] = \mathbb{E}[XY]\text{]} \\
&= \mathbb{E}\left[\left(\frac{1}{\sqrt{n_0}} \sum_{i=1}^{n_0} W_{ki}^{(0)} x_i + \beta b_k^{(0)}\right)\left(\frac{1}{\sqrt{n_0}} \sum_{i=1}^{n_0} W_{ki}^{(0)} x'_i + \beta b_k^{(0)}\right)\right] \\
&= \frac{1}{n_0} \mathbb{E}\left[\left(W_{k,\bullet}^{(0)}\right)^T x^T(x') W_{k,\bullet}^{(0)}\right] + \frac{1}{\sqrt{n_0}} 2\beta \mathbb{E}\left[\left(W_{k,\bullet}^{(0)}\right)^T x^T b_k^{(0)}\right] + \beta^2 \mathbb{E}\left[b_k^{(0)} b_k^{(0)}\right] \\
&= \frac{1}{n_0} x^T(x') + \beta^2
\end{aligned}$$

where in the last passage we derived the expectation considering that x, x' are fixed and using the iid discussion of $\theta = \{W^{(0)}, b^{(0)}\}$ above the calculations.

(🚗 inductive hypothesis) assume the claim is true $\forall L$.

(🐦‍⬛ inductive step) the key to proving the hypothesis for $L+1$ is recognizing that a network of depth $L+1$ can be decomposed into:

- a subnetwork of depth L mapping onto the subfunctional space $f \in \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ via the preactivations $\widetilde{\alpha}_i^{(L)}$
- elementwise activations σ

- a random affine map as in  , but from $\mathbb{R}^{n_L}$ to $\mathbb{R}^{n_{L+1}}$

from the inductive hypothesis of  we obtain that the subnetwork at the sequential limit of hidden neurons has preactivations $\{\widetilde{\alpha}_i^{(L)}\}_{i=1}^{n_L}$ with the form of iid Gaussian processes with $\Sigma^{(L)}$ as claimed. The output of the network is instead after the other two steps:

$$\begin{aligned}
f_\theta &= \widetilde{\alpha}^{(L+1)} \in \mathbb{R}^{n_{L+1}} \\
&= \frac{1}{\sqrt{n_L}} W^{(L)} \alpha^{(L)} + \beta b^{(L)} \\
&= \frac{1}{\sqrt{n_L}} W^{(L)} \sigma\left(\widetilde{\alpha}^{(L)}\right) + \beta b^{(L)} \\
\implies \widetilde{\alpha}_k^{(L+1)}(x) &= \frac{1}{\sqrt{n_L}} \sum_{i=1}^{n_L} W_{ki}^{(L)} \sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right) + \beta b_k^{(L)} \qquad k \in \{1, \dots, n_{L+1}\} \\
&= \frac{1}{\sqrt{n_L}} \left(W_{k,\bullet}^{(L)}\right)^T \sigma \oplus \widetilde{\alpha}^{(L)}(x) + \beta b_k^{(L)}
\end{aligned}$$

where $\oplus$ means element-wise application of the map σ to a vector. To check that f_θ is a Gaussian process, just notice that it is a combination of Gaussian processes by the inductive assumption. To see if the claimed form of $\Sigma^{(L+1)}$ is verified, observe that the expectation of f_θ is the sum of the inner expectations in the sum, which is in principle easier to check. Easily see that:

$$\begin{aligned}
\mathbb{E}\left[W_{ki}^{(L)} \sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right)\right] &= \mathbb{E}\left[W_{ki}^{(L)}\right] \mathbb{E}\left[\sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right)\right] && \text{independence} \\
&= \mu_{\theta_p} \mathbb{E}\left[\sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right)\right] && \text{identical} \\
&= 0 && \mu_{\theta_p} = 0
\end{aligned}$$

By the inductive assumption the variance of the sums is the sum of the variances. Given the previous calculations, the null expectation lets us conclude that:

$$\begin{aligned}
\texttt{Var}\left[W_{ki}^{(L)} \sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right)\right] &= \mathbb{E}\left[\left(W_{ki}^{(L)} \sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right)\right)^2\right] \\
&= \mathbb{E}\left[\left(W_{ki}^{(L)}\right)^2\right] \mathbb{E}\left[\left(\sigma\left(\widetilde{\alpha}_i^{(L)}(x)\right)\right)^2\right] && \text{independence} \\
&= \sigma_{\theta_p}^2 \Sigma^{(L)}(x, x) && \text{identical} \\
&= \Sigma^{(L)}(x, x) && \sigma_{\theta_p}^2 = 1
\end{aligned}$$

Similarly to the base case in  , it holds that the candidate covariance at finite width is:

$$\begin{aligned}
\widetilde{\Sigma}^{(L)}(x, x') &= \mathbb{E}\left[\widetilde{\alpha}_k^{(L+1)}(x) \widetilde{\alpha}^{(L+1)}(x')\right] \\
&= \frac{1}{n_L} \sigma\left(\widetilde{\alpha}^{(L)}(x; \theta)\right) \sigma\left(\widetilde{\alpha}^{(L)}(x'; \theta)\right) + \beta^2 \\
&= \frac{1}{n_L} \alpha^{(L)}(x; \theta) \alpha^{(L)}(x'; \theta) + \beta^2
\end{aligned}$$

Recall that, subject to sequential limit of $n_1 \to \infty, \dots, n_{L-1} \to \infty$, $\widetilde{\alpha}^{(L)}$ is a Gaussian process with covariance function $\Sigma^{(L)}$ by the inductive hypothesis. Then, letting $n_L \to \infty$ as well, we

have that by the law of large numbers the red term will tend in probability law to the expectation of a sampling from such gaussian process. Namely:

$$\frac{1}{n_L}\alpha^{(L)}(x;\theta)\alpha^{(L)}(x';\theta) \underset{n_L\to\infty}{\overset{\mathcal{P}}{\to}} \mathbb{E}_{f\sim\mathcal{N}(0,\Lambda^{(L)})}\left[\sigma(f(x))\sigma(f(x'))\right]$$

and combining this with the blue term we obtain the claim:

$$\widetilde{\Sigma}^{(L+1)}(x,x') \underset{n_L\to\infty}{\overset{\mathcal{P}}{\to}} \Sigma^{(L+1)}(x,x') = \mathbb{E}_{f\sim\mathcal{N}(0,\Lambda^{(L)})}\left[\sigma(f(x))\sigma(f(x'))\right] + \beta^2$$

□

Observation III.1.3 (Gaussian Process Existance)**.** *Throughout the proof we focused on a realization of the $\mathcal{GP}$ process at $f(x), f(x')$, which is only a Gaussian vector. This ensures existance without the need to discuss the validity of the covariance Σ [JGH20].*

With neurons sequentially diverging, it is also possible to prove that the Neural Tangent Kernel converges to a deterministic limit, and is not anymore random at initialization as mentioned in the finite sample size case of Observation II.1.5.

Theorem III.1.4 (Initial NTK stability at the limit)**.** *For a network with:*

- *L layers*
- *σ Lipschitz non-linearities*

it holds that:

$$\lim_{n_{L-1}\to\infty}\cdots\lim_{n_1\to\infty}\boldsymbol{\Theta^{(L)}} = \boldsymbol{\Theta^{(L)}_\infty}\otimes Id_{n_L}$$

where the limiting kernel is defined on a single output neuron as:

$$\boldsymbol{\Theta^{(L)}_\infty} : \mathbb{R}^{n_0}\times\mathbb{R}^{n_0}\to\mathbb{R}$$

and the Id_{n_L} matrix is used to extend it to all n_L neurons with the trivial multiplication dimensional kernel to obtain a Kernel that maps to $\mathbb{R}^{n_L}\times\mathbb{R}^{n_L}$.
The form of $\boldsymbol{\Theta^{(L)}_\infty}$ is described recursively as:

$$\begin{aligned}\boldsymbol{\Theta^{(1)}_\infty}(x,x') &= \Sigma^{(1)}(x,x')\\ \boldsymbol{\Theta^{(L+1)}_\infty}(x,x') &= \boldsymbol{\Theta^{(L)}_\infty}(x,x')\dot{\Sigma}^{(L+1)}(x,x') + \Sigma^{(L+1)}(x,x')\end{aligned}$$

where:

- *$\Sigma^{(L)}$ are the covariances of the Gaussian processes of the entries of the network function from Proposition III.1.2*
- *$\dot{\Sigma}^{(L+1)} := \mathbb{E}_{f\sim\mathcal{N}(0,\Sigma^{(L)})}\left[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))\right]$*

Remark. *While $\dot{\Sigma}$ is a notation choice and not a derivative, $\dot{\sigma}$ is a derivative.*

Remark. *By Rademacher Theorem (Thm. A.2.7) the Lipschitz non-linearity σ is differentiable almost everywhere.*

Remark. *The limit is completely determined by the non-linearity σ, the depth L and the variance of the initialization*

Proof. (**strategy**) we use an induction argument in L.

(**base case** $L = 1$) no hidden layers mean that no limit can be taken, as we only have the input and output layers (respectively, 0^{th} and N^{th}), which are fixed and depend on $\mathcal{D}$. The network function is:

$$\begin{aligned} f_\theta(x) &= \widetilde{\alpha}^{(1)}(x) \\ &= \frac{1}{\sqrt{n_0}} W^{(0)} x + \beta b^{(0)} \end{aligned}$$

Recall the construction of Equation II.1.3. Omitting the layer index L since $L = 1$ and specifying two inputs x, x' with θ now as a parameter we get:

$$\mathbf{\Theta}_{kk'}(x, x'; \theta) = \left(\frac{\partial f_{\theta,k}(x')}{\partial W^{(0)}}\right)^T \left(\frac{\partial f_{\theta,k'}(x)}{\partial W^{(0)}}\right) + \left(\frac{\partial f_{\theta,k}(x')}{\partial b^{(0)}}\right)^T \left(\frac{\partial f_{\theta,k'}(x)}{\partial b^{(0)}}\right)$$

We could also express this derivative directly concluding with the base case, but opt to make the argument even more specific to familiarize with the formulas. For a single entry k, k' the NTK is:

$$\begin{aligned} \mathbf{\Theta}_{kk'}(x, x') &= \sum_{p=1}^{P} \partial_{\theta_p} F_k^{(1)}(\theta, x) \partial_{\theta_p} F_{k'}^{(1)}(x, \theta) && \text{Eqn. II.1.4} \\ &= \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} \overbrace{\frac{1}{n_0} x_i x_i' \delta_{jk} \delta_{jk'}}^{\partial_{W_{ij}^{(0)}} f_k(x) \partial_{W_{ij}^{(0)}} f_{k'}(x')} + \sum_{j=1}^{n_1} \overbrace{\beta^2 \delta_{jk} \delta_{jk'}}^{\partial_{b_j^{(0)}} f_k(x) \partial_{b_j^{(0)}} f_{k'}(x')} && \delta \text{ are dirac masses} \\ &= \frac{1}{n_0} x^T x' \delta_{kk'} + \beta^2 \delta_{kk'} \\ &= \Sigma^{(1)}(x, x') \delta_{kk'} \\ \implies \mathbf{\Theta}(x, x') &= \Sigma^{(1)}(x, x') \otimes I_{n_1} && \text{by } k, k' \text{ arbitrariness} \end{aligned} \tag{III.1.5}$$

(**inductive hypothesis**) assume the claim is true $\forall L$.

(**inductive step**) Like Proposition III.1.2 recognize that a network of depth $L + 1$ can be decomposed into:

- a subnetwork of depth L mapping onto the subfunctional space $f \in \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ via the preactivations $\widetilde{\alpha}_i^{(L)}$
- elementwise activations σ
- a random affine map from $\mathbb{R}^{n_L}$ to $\mathbb{R}^{n_{L+1}}$

Additionally, split $\theta \in \mathbb{R}^P$ as follows:

$$\mathbb{R}^{\widetilde{P}} \ni \widetilde{\theta} = \left\{W^{(\ell)}, b^{(\ell)}\right\}_{\ell=1}^{L} \quad \left\{(W^{(L+1)}, b^{(L+1)}\right\} \quad \theta = \left\{\widetilde{\theta}, (W^{(L+1)}, b^{(L+1)}\right\} \in \mathbb{R}^P$$

Consider:

- the L-deep subnetwork

- single entries to avoid the tensor product, namely preactivations $\widetilde{\alpha}_i^{(L)}$ for $i \in \{1, \dots, n_L\}$. Notice that we can do this since the inductive hypothesis states that the n_L neurons are independent (focus on $\otimes Id_{n_L}$ to understand this)
- the set of parameters $\widetilde{\theta}$

Then:

- by Proposition III.1.2 $\widetilde{\alpha}_i^{(L)}$ are iid Gaussian processes with covariance function $\Sigma^{(L)}$
- by the inductive hypothesis in 🚗 the element-wise NTK converges, irrespectively of ii' to:

$$\begin{aligned}
\boldsymbol{\Theta^{(L)}}_{i,i'}(x, x'; \widetilde{\theta}) &= \left(\partial_{\widetilde{\theta}} F_i^{(L)}(x; \theta)\right)^T \left(\partial_{\widetilde{\theta}} F_i^{(L)}(x'; \theta)\right) \\
&= \left(\partial_{\widetilde{\theta}} \widetilde{\alpha}_i^{(L)}(x; \theta)\right)^T \left(\partial_{\widetilde{\theta}} \widetilde{\alpha}_i^{(L)}(x'; \theta)\right) \\
&\to \boldsymbol{\Theta_\infty^{(L)}}(x, x') \qquad \text{as } n_1 \to \infty, \dots, n_L \to \infty \quad \text{(III.1.6)}
\end{aligned}$$

Observe that for the $L+1$ level network we have a network function:

$$f_\theta(x) = \widetilde{\alpha}^{(L+1)}(x; \theta) = \frac{1}{\sqrt{n_L}} W^{(L)} \sigma\left(\widetilde{\alpha}^{(L)}(x)\right) + \beta b^{(L)}$$

we could split its NTK into sums of derivatives by linearity of the derivatives:

$$\partial_\theta f_\theta(x) = \partial_{\widetilde{\theta}} f_\theta(x) + \partial_{W^{(L)}} f_\theta(x) + \partial_{b^{(L)}} f_\theta(x)$$

so that the NTK takes form:

$$\begin{aligned}
\boldsymbol{\Theta^{(L+1)}}_{kk'} &= \left(\partial_\theta f_{\theta,k}\right)^T \partial_\theta f_{\theta,k'} \\
&= \left[\partial_{\widetilde{\theta}} f_{\theta,k}\right]^T \partial_{\widetilde{\theta}} f_{\theta,k'} + \left[\partial_{W^{(L)}} f_{\theta,k}\right]^T \partial_{W^{(L)}} f_{\theta,k'} + \left[\partial_{b^{(L)}} f_{\theta,k}\right]^T \partial_{b^{(L)}} f_{\theta,k'} \qquad \text{(III.1.7)}
\end{aligned}$$

the aim is inspecting the blue and red terms separately for clearness. Coincidentally, we will derive their contribution to the NTK.

(🔏 🦚 subpoint, blue) This is the easy part. By the form of the network function we easily conclude that:

$$\partial_{W^{(L)}} f_\theta(x) = \frac{1}{\sqrt{n_L}} \sigma\left(\widetilde{\alpha}^{(L)}(x)\right)^T \in \mathbb{R}^{1 \times n_L} \qquad \partial_{b^{(L)}} f_\theta(x) = \beta \in \mathbb{R}$$

the sum of their kernel contributions is:

$$\left[\partial_{W^{(L)}} f_\theta(x)\right]^T \partial_{W^{(L)}} f_\theta(x') + \left[\partial_{b^{(L)}} f_\theta(x)\right]^T \partial_{b^{(L)}} f_\theta(x') = \frac{1}{n_L} \sigma\left(\widetilde{\alpha}^{(L)}(x)\right) \sigma\left(\widetilde{\alpha}^{(L)}(x')\right)^T + \beta^2$$

By Proposition III.1.2, an abitrary entry k, k' of the blue part is such that:

$$blue_{kk'}(x, x') \to \Sigma^{(L+1)}(x, x') \delta_{kk'} \quad \text{as } n_1 \to \infty, \dots, n_L \to \infty \qquad \text{(III.1.8)}$$

(🐦 **subpoint, red**) The more articulate term can be expressed by using the chain rule:

$$\begin{aligned}
\partial_{\widetilde{\theta}} f_\theta(x) &= \frac{1}{\sqrt{n_L}} \left[\partial_{\widetilde{\theta}} \sigma\left(\widetilde{\alpha}^{(L)}(x)\right)\right]^T \left(W^{(L)}\right)^T \\
&= \frac{1}{\sqrt{n_L}} \begin{bmatrix} \dot{\sigma}\left(\widetilde{\alpha}_1^{(L)}(x)\right) \dfrac{\partial \widetilde{\alpha}_1^{(L)}}{\partial \widetilde{\theta}_1} & \cdots & \dot{\sigma}\left(\widetilde{\alpha}_{n_L}^{(L)}(x)\right) \dfrac{\partial \widetilde{\alpha}_{n_L}^{(L)}}{\partial \widetilde{\theta}_1} \\ \vdots & & \vdots \\ \dot{\sigma}\left(\widetilde{\alpha}_1^{(L)}(x)\right) \dfrac{\partial \widetilde{\alpha}_1^{(L)}}{\partial \widetilde{\theta}_{\widetilde{P}}} & \cdots & \dot{\sigma}\left(\widetilde{\alpha}_{n_L}^{(L)}(x)\right) \dfrac{\partial \widetilde{\alpha}_{n_L}^{(L)}}{\partial \widetilde{\theta}_{\widetilde{P}}} \end{bmatrix} \left(W^{(L)}\right)^T \\
&= \frac{1}{\sqrt{n_L}} \left[\left(\partial_{\widetilde{\theta}} \widetilde{\alpha}^{(L)}\right) \dot{\sigma} \oplus \widetilde{\alpha}^{(L)}\right] \left(W^{(L)}\right)^T \in \mathbb{R}^{\widetilde{P} \times n_{L+1}}
\end{aligned}$$

where the last form is in compact notation, with $\oplus$ meaning element wise application. For a single parameter entry $p \in \{1, \dots, \widetilde{P}\}$ and a single neuron $k \in \{1, \dots, n_{L+1}\}$ we find that:

$$\partial_{\widetilde{\theta}_p} f_{\theta,k}(x) = \frac{1}{\sqrt{n_L}} \sum_{i=1}^{n_L} \partial_{\widetilde{\theta}_p} \widetilde{\alpha}_i^{(L)}(x;\theta) \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) W_{ki}^{(L)}$$

We could write the k, k' entry-wise contribution of these terms to the NTK as:

$$\left[\partial_\theta f_{\theta,k} \partial_\theta f_{\theta,k'}\right](x,x') = \frac{1}{n_L} \sum_{i=1}^{n_L} \sum_{i'=1}^{n_L} \underbrace{\sum_{p=1}^{\widetilde{P}} \left[\partial_{\theta_p} \widetilde{\alpha}_i^{(L)}(x;\theta) \partial_{\theta_p} \widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right]}_{=\boldsymbol{\Theta}^{(L)}{}_{i,i'}(x,x';\theta)} \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right) W_{ki}^{(L)} W_{ik'}^{(L)}$$

Where in the sequential limit the braced term is, by the reasoning of Equation III.1.6, convergent in law to the NTK at the subnetwork $\boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}$. Thus, we first let $n_1 \to \infty, \dots, n_{L-1} \to \infty$, and eventually use the law of large numbers for $n_L \to \infty$. This entails:

$$\begin{aligned}
&\frac{1}{n_L} \sum_{i=1}^{n_L} \sum_{i'=1}^{n_L} \boldsymbol{\Theta}_{\boldsymbol{i,i'}}^{(\boldsymbol{L})}(x,x';\theta) \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right) W_{ki}^{(L)} W_{ik'}^{(L)} \\
&[\{n_\ell\} = \{n_1, \dots, n_{L-1}\} \text{ sequentially}] \\
&\overset{\{n_\ell\} \to \infty}{\to} \frac{1}{n_L} \sum_{i=1}^{n_L} \sum_{i'=1}^{n_L} \boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}(x,x') \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right) W_{ki}^{(L)} W_{ik'}^{(L)} \\
&= \boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}(x,x') \frac{1}{n_L} \sum_{i=1}^{n_L} \sum_{i'=1}^{n_L} \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right) W_{ki}^{(L)} W_{ik'}^{(L)} \\
&[\text{use Law of Large Numbers}] \\
&\overset{n_L \to \infty}{\to} \boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}(x,x') \mathbb{E}\left[\frac{1}{n_L} \sum_{i=1}^{n_L} \sum_{i'=1}^{n_L} \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right) W_{ki}^{(L)} W_{ik'}^{(L)}\right] \\
&\left[\text{use independence and } \mathbb{E}\left[\left(W^{(L)}\right)^T \left(W^{(L)}\right)\right] = \texttt{Var}\left[W^{(L)}\right] = 1\right] \\
&= \boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}(x,x') \mathbb{E}\left[\dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right)\right] \underbrace{\mathbb{E}\left[W_{ki}^{(L)} W_{ik'}^{(L)}\right]}_{\delta_{kk'}} \\
&= \boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}(x,x') \mathbb{E}_{f \sim \mathcal{N}(0,\Sigma^{(L)})}\left[\dot{\sigma}\left(f(x)\right) \dot{\sigma}\left(f(x')\right)\right] \delta_{kk'} \\
&= \boldsymbol{\Theta}_{\boldsymbol{\infty}}^{(\boldsymbol{L})}(x,x') \dot{\Sigma}^{(L+1)}(x,x') \delta_{kk'}
\end{aligned}$$

where we recovered the notation of the claim in the last equality. It is clear that with the colors used:

$$red_{kk'}(x,x') \to \boldsymbol{\Theta}^{(\boldsymbol{L})}_{\boldsymbol{\infty}}(x,x')\dot{\Sigma}^{(L+1)}(x,x')\delta_{kk'} \quad \text{as } n_1 \to \infty, \dots, n_L \to \infty \tag{III.1.9}$$

(**subpoint, recap of inductive step)** we have shown that the blue and red terms at the limit have a specific form, in line with the claim of the Theorem. The induction is completed by combining the results of Equations III.1.7, III.1.8, III.1.9. Indeed the $\delta_{k,k'}$ translate to the tensor product extension in n_{L+1}. To see these, recollect the Equations cited and see that:

$$\boldsymbol{\Theta}^{(\boldsymbol{L+1})}(x,x') = \left[\underbrace{\boldsymbol{\Theta}^{(\boldsymbol{L})}_{\boldsymbol{\infty}}(x,x')\dot{\Sigma}^{(L+1)}(x,x') + \Sigma^{(L)}(x,x')}_{=\boldsymbol{\Theta}^{(\boldsymbol{L+1})}_{\boldsymbol{\infty}}} \right] \otimes Id_{n_{L+1}} \in \mathbb{R}^{n_{L+1}\times n_{L+1}}$$

and the recursive claim is verified. □

Observation III.1.10 (About the NTK at initialization)**.** *The kernel we found is deterministic and diagonal in the $k = 1, \dots, n_L$ neurons of the output. This means that the n_L sized output is equivalent to n_L outputs stacked and independent. With the next result, we will see that this holds also during training, and so training a vector output ANN is equal to training many scalar output ANNs in parallel.*

III.2 Sequential Kernel Convergence

The authors propose a more general result in which the learning setup has an arbitrary training direction $d_t \in \mathcal{F}$. The parameter update in this subsection is thus lifted to the form:

$$\partial_t \theta_p(t) = \left\langle \partial_{\theta_p} F^{(L)}(\theta(t)), d_t \right\rangle_{p^{in}}$$

to recover the specific case of gradient descent we specifically choose $d_t = -d|_{f(\theta(t))} = -\frac{\delta C}{\delta f(\theta(t))} \in \mathcal{F}$.

The width of the result extends to cases in which training is not dependent on the network we are optimizing. A clarifying example is that of Generative Adversarial Networks (see [Goo+14] for the original paper). The intuition is that up to weaker assumptions it is still possible to conclude that the NTK is asymptotically constant during training, and thus is **a good approximation of the dynamics** of learning recovered in **closed form**. Weights θ_p are still initialized as standard Gaussians.

To assess convergence of the NTK in time, we need two objects:

- a proper criterion (Def. III.2.1)
- a technical Lemma (Lem. III.2.2)

Definition III.2.1 (Kernel Operator Norm)**.** *Given a Kernel $\boldsymbol{K}$ (Def. I.3.4) define its operator norm (Def. A.1.8) with the classical method with respect to the input distribution which draws*

from the domain of the functions that the kernel operates on:

$$\begin{aligned}\|\boldsymbol{K}\|_{op} &\coloneqq \max_{f\in\mathcal{F},\|f\|_{p^{in}}\leq 1}\|f\|_{\boldsymbol{K}}\\ &= \max_{f\in\mathcal{F},\|f\|_{p^{in}}\leq 1}\sqrt{\mathbb{E}_{x,x'\sim p^{in},x\perp x'}\left[f(x)^T\boldsymbol{K}(x,x')f(x')\right]} && \textit{Eqn. I.3.5}\\ &= \max_{\|f\|_{p^{in}}\leq 1}\sqrt{\mathbb{E}_{x,x'}\left[f(x)^T\boldsymbol{K}(x,x')f(x')\right]} && \textit{shorter notation}\end{aligned}$$

Remark. *By Assumption I.2.6, p^{in} is the empirical distribution of the features in $\mathcal{D}$. This means that $\|\boldsymbol{K}\|_{op}$ is the largest eigenvalue of the Gram matrix $(\boldsymbol{K}_{kk'}(x_i,x_j))_{i,j=0,\,k,k'=0}^{i,j=N-1,\,k,k'=n_{L-1}}$ (Def. A.2.12). We take this result for granted, but stress that clearly by the bound on the norm of f and the direct dependence on the dataset induced distribution the Gram matrix will be the only factor influencing the operator norm. By such matrix being symmetric, we obtain that the norm is bounded by the largest eigenvalue. This result is formally proved in [SC04](Section 3), along with many other important properties of Kernels.*

Remark. *A more direct and still important implication of Assumption I.2.6 on the input distribution is that convergence in operator norm is equivalent to convergence on all the individuals sampled. Namely, given a sequence of kernels and a target:*

$$\left\|\boldsymbol{K}^{(\boldsymbol{n})}-\boldsymbol{K}\right\|_{op}\overset{n\to\infty}{\to}0 \iff \boldsymbol{K}^{(\boldsymbol{n})}(x,x')-\boldsymbol{K}(x,x')\overset{n\to\infty}{\to}0 \quad \forall x,x'\in\mathcal{D}$$

Lemma III.2.2 (Infinite-width weights are static)**.** *For a network with:*

- *$L+1$ layers*
- *σ Lipschitz non-linearities (i.e. part of the original Assumption on non-linearities I.2.7)*

It holds in probability law that for all T the weights do not move across the learning dynamics:

$$\lim_{n_L\to\infty}\cdots\lim_{n_1\to\infty}\sup_{t\in[0,T]}\left\|\frac{1}{\sqrt{n_\ell}}\left(W^{(\ell)}(t)-W^{\ell}(0)\right)\right\|_{op}=0 \quad \forall\ell\in 1,\dots,L$$

Remark. *This interesting result is linked with the notion of Lazy training we will discuss in IV.1.*

Proof. The authors give a proof, for the sake of time, we just reference it [JGH20]. □

Theorem III.2.3 (Preserved NTK stability at the limit)**.** *For a network with:*

- *L layers*
- *σ Lipschitz non-linearities (i.e. the original Assumption on non-linearities I.2.7)*

it holds that for any T satisfying

$$\int_0^T\|d_t\|_{p^{in}}\,dt<\infty \quad \textit{stochastically}$$

we have that in the sequential limit $n_1\to\infty,\dots,n_{L-1}\to\infty$ the kernel is:

$$\boldsymbol{\Theta}^{(\boldsymbol{L})}(t)\overset{\{n_\ell\}\to\infty}{\underset{t\in[0,T]}{\rightrightarrows}}\boldsymbol{\Theta}^{(\boldsymbol{L})}_{\boldsymbol{\infty}}\otimes Id_{n_L}$$

where the symbol $\overset{\{n_\ell\}\to\infty}{\underset{t\in[0,T]}{\rightrightarrows}}$ means:

- *in the sequential limit of the hidden neurons*
- *uniformly in $t \in [0, T]$ (Def. A.2.8)*

With this premise, we could then establish that the network function evolves according to the differential equation:

$$\partial_t f_{\theta(t)} = -\Phi_{\mathbf{\Theta}_{\infty}^{(L)} \otimes Id_{n_L}} \left(\langle d_t, \cdot \rangle_{p^{in}} \right)$$

Remark. *The claim "asymptotic constant during training" [JGH20] is exactly answered by the uniform convergence over stochastically bounded intervals. The evolution across time of the kernel at the diverging limit is described by a single constant kernel. This resembles the case of Subsection I.4 of approximating by random functions.*

Proof. (𓂀 **strategy)** We use an induction argument in the number of layers.

(𓆑 **base case** $L = 1$**)** with no hidden layers, the NTK is as in Equation III.1.5, namely:

$$\mathbf{\Theta}(x, x') = \Sigma^{(1)}(x, x') = \frac{1}{n_0} x^T x' + \beta^2$$

with the formulation completely independent of θ and thus constant during training. Notice however that θ changes over time. It is just that the difference does not fluctuate depending on its values.

(𓂋 **inductive hypothesis)** assume the claim is true $\forall L$.

(𓅞 **inductive step)** For an $L+1$-deep network, as in the previous proofs, split the problem into a combination of:

- an L-deep subnetwork with parameters $\widetilde{\theta}$
- the last layer with the remaining parameters $W^{(L)}, b^{(L)}$

For $i = 1, \dots, n_L$, denoting $\widetilde{\alpha}_i^{(L)}(t) \coloneqq \widetilde{\alpha}_i^{(L)}(\cdot; \theta(t))$ for simplicity, it holds that the subnetwork follows the training direction:

$$d'_t = \dot{\sigma}\left(\widetilde{\alpha}^{(L)}(t)\right) \left(\frac{1}{\sqrt{n_L}} W^{(L)}(t)\right)^T d_t \tag{III.2.4}$$

Notice that with this statement we mean that the whole subnetwork, is tweaked to follow this training direction once confronted with the parameters of the last layer. In some sense:

- d_t is the old training direction, stochastically bounded by the inductive hypothesis
- d'_t is the training direction of all the layers but one with respect to the last layer

We then need to inspect when d'_t is stochastically bounded in the sequential limit.

(𓂀 𓅞 **subpoint, stochastic boundedness)** one of the hypothesis of the Theorem on the structure of the network is that the non-linearities are Lipschitz. Let σ be specifically c-Lipschitz,

so that Lemma A.2.2 holds and $|\dot{\sigma}| \leq c$. Then:

$$\begin{aligned}
\left\|d_t'\right\|_{p^{in}} &= \left\|\dot{\sigma}\left(\widetilde{\alpha}^{(L)}(t)\right)\left(\frac{1}{\sqrt{n_L}}W^{(L)}(t)\right)^T d_t\right\|_{p^{in}} \\
&\leq \left\|\dot{\sigma}\left(\widetilde{\alpha}^{(L)}(t)\right)\right\|_{p^{in}} \left\|\left(\frac{1}{\sqrt{n_L}}W^{(L)}(t)\right)^T\right\|_{op} \|d_t\|_{p^{in}} \quad \text{Def. A.1.8 \& Cauchy-Schwarz} \\
&\leq c\left\|\frac{1}{\sqrt{n_L}}W^{(L)}(t)\right\|_{op} \|d_t\|_{p^{in}} \qquad\qquad \|A\|_{op} = \left\|A^T\right\|_{op}
\end{aligned}$$

In the sequential limit, Lemma III.2.2 allows us to use $W^{(L)}(0)$ instead of $W^{(L)}$, which means that we can inspect:

$$\mathbb{R}^{n_{L+1}\times n_L} \ni \frac{1}{\sqrt{n_L}}W^{(L)}(0) = \begin{bmatrix} — & \frac{1}{\sqrt{n_L}}W^{(L)}_{1,\bullet}(0) & — \\ \vdots & \vdots & \vdots \\ — & \frac{1}{\sqrt{n_L}}W^{(L)}_{n_{L+1},\bullet}(0) & — \end{bmatrix}$$

Using the law of large numbers on the norms of the n_L dimensional rows, their norm is bounded. It follows that the operator norm of $W^{(L)}(0)$ is as well (Lemma A.1.9).

(subpoint, setup for inductive step) having boundedness of the direction from , we can eventually say that by the inductive hypothesis of the dynamics of the L-deep network are described in the sequential limit $\lim_{n_L\to\infty}\cdots\lim_{n_1\to\infty}$ by:

$$\partial_t\widetilde{\alpha}_i^{(L)}(t) = \frac{1}{\sqrt{n_L}}\Phi_{\boldsymbol{\Theta}_\infty^{(L)}}\left(\left\langle \underbrace{\dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(t)\right)\left(W_{i,\bullet}^{(L)}(t)\right)^T d_t}_{=d_t'}, \cdot \right\rangle_{p^{in}}\right) \quad i = 1,\ldots,n_{L+1} \qquad \text{(III.2.5)}$$

Notice that here we still have a $\mathcal{O}\left(n_L^{-\frac{1}{2}}\right)$ factor in front since we let all the neurons diverge but the last layer.

Discarding $b^{(L)}$ which evolves with a β rate, the entries of the weight matrix at the last layer evolve according to:

$$\partial_t W_{ij}^{(L)}(t) = \frac{1}{\sqrt{n_L}}\left\langle \sigma\left(\widetilde{\alpha}_i^{(L)}(t)\right), d_{t,j}\right\rangle_{p^{in}} = \frac{1}{\sqrt{n_L}}\left\langle \alpha_i^{(L)}(t), d_{t,j}\right\rangle_{p^{in}} \qquad \text{(III.2.6)}$$

We wish to bound the variation of weights column weights $W_{i,\bullet}^{(L)}(t)$ and individual preactivations $\widetilde{\alpha}_i^{(L)}(t)$ at the last layer. The former is in L^2-norm, the latter in p^{in} norm. This makes sense

since one is a matrix and the other is a function on inputs. The weights are such that:

$$\begin{aligned}
\partial_t \left\| W_{i,\bullet}^{(L)}(t) - W_{i,\bullet}^{(L)}(0) \right\|_{L^2} &\leq \left\| \partial_t W_{i,\bullet}^{(L)}(t) - \partial_t W_{i,\bullet}^{(L)}(0) \right\|_{L^2} && \text{Lemma A.2.15} \\
&= \frac{1}{\sqrt{n_L}} \left\| \begin{bmatrix} \left\langle \alpha_i^{(L)}(t), d_{t,1} \right\rangle_{p^{in}} \\ \vdots \\ \left\langle \alpha_i^{(L)}(t), d_{t,n_L} \right\rangle_{p^{in}} \end{bmatrix} \right\|_{L^2} && \text{Eqn. III.2.6} \\
&\leq \frac{1}{\sqrt{n_L}} \left\| \begin{bmatrix} \left\| \alpha_i^{(L)}(t) \right\|_{p^{in}} \left\| d_{t,1} \right\|_{p^{in}} \\ \vdots \\ \left\| \alpha_i^{(L)}(t) \right\|_{p^{in}} \left\| d_{t,n_L} \right\|_{p^{in}} \end{bmatrix} \right\|_{L^2} && \text{Cauchy-Schwarz} \\
&= \frac{1}{\sqrt{n_L}} \left\| \alpha_i^{(L)}(t) \right\|_{p^{in}} \left\| \begin{bmatrix} \left\| d_{t,1} \right\|_{p^{in}} \\ \vdots \\ \left\| d_{t,n_L} \right\|_{p^{in}} \end{bmatrix} \right\|_{L^2} && \text{see below} \\
&= \frac{1}{\sqrt{n_L}} \left\| \alpha_i^{(L)}(t) \right\|_{p^{in}} \left\| d_t \right\|_{p^{in}}
\end{aligned}$$

where we first the simple fact that the L^2 norm is smaller than the L^1 norm, and we are doing a norm of norms (positive). Namely:

$$\sqrt{\sum |q_i|^2} \leq \sqrt{\left(\sum q_i\right)^2} = \left| \sum q_i \right| \overset{q_i \geq 0 \forall i}{=} \sum q_i$$

and then the intuitive fact that the empirical norm of all the dimensions is the empirical norm of the vector. This concludes the weights bound part.
For the preactivations instead we first observe quickly that by the discussion of Subsection I.3.1:

$$\left\| \Phi_{\boldsymbol{\Theta}_\infty^{(L)}} \right\|_{op} = \left\| \boldsymbol{\Theta}_\infty^{(L)} \right\|_{op}$$

which allows us to say that:

$$\begin{aligned}
\partial_t \left\| \widetilde{\alpha}_i^{(L)}(t) - \widetilde{\alpha}_i^{(L)}(0) \right\|_{p^{in}} &\leq \left\| \partial_t \widetilde{\alpha}_i^{(L)}(t) - \partial_t \widetilde{\alpha}_i^{(L)}(0) \right\|_{p^{in}} && \text{as before} \\
&= \left\| \partial_t \widetilde{\alpha}^{(L)}(t) \right\|_{p^{in}} \\
&= \frac{1}{\sqrt{n_L}} \left\| \Phi_{\boldsymbol{\Theta}_\infty^{(L)}} \left(\left\langle d_t', \cdot \right\rangle_{p^{in}} \right) \right\|_{p^{in}} && d_t' \text{ as in Eqn. III.2.5} \\
&\leq \frac{1}{\sqrt{n_L}} \left\| \Phi_{\boldsymbol{\Theta}_\infty^{(L)}} \right\|_{op} \left\| d_t' \right\|_{p^{in}} \\
&\leq \frac{1}{\sqrt{n_L}} \left\| \boldsymbol{\Theta}_\infty^{(L)} \right\|_{op} \left\| \dot{\sigma} \left(\widetilde{\alpha}_i^{(L)}(t) \right) \right\|_{\infty} \left\| W_{i,\bullet}^{(L)} \right\|_{L^2} \left\| d_t \right\|_{p^{in}}
\end{aligned}$$

where in the last passage we just unrolled the content of Equation III.2.4 with the proper norms for our purpose[5].

[5] the norm on the derivative of the sigmoid is the classic sup norm for which $\|f\|_\infty = \sup_x |f(x)|$

(**subpoint, a joint quantity**) to bound together the two objects a function that combinates them is introduced:

$$A(t) \coloneqq \left\|\alpha_i^{(L)}(0)\right\|_{p^{in}} + c\left\|\widetilde{\alpha}_i^{(L)}(t) - \widetilde{\alpha}_i^{(L)}(0)\right\|_{p^{in}} + \left\|W_{i,\bullet}^{(L)}(0)\right\|_{L^2} + \left\|W_{i,\bullet}^{(L)}(t) - W_{i,\bullet}^{(L)}(0)\right\|_{L^2}$$

It holds that by the bounds on the weights and preactivations that:

$$\partial_t A(t) \leq \underbrace{\frac{1}{\sqrt{n_L}}\left\|\alpha_i^{(L)}(t)\right\|_{p^{in}} \|d_t\|_{p^{in}}}_{\text{weights inequality}} + \underbrace{c\frac{1}{\sqrt{n_L}}\left\|\boldsymbol{\Theta}_{\infty}^{(\boldsymbol{L})}\right\|_{op}\left\|\dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(t)\right)\right\|_{\infty}\left\|W_{i,\bullet}^{(L)}\right\|_{L^2}\|d_t\|_{p^{in}}}_{\text{preactivations inequality}}$$

$$= \frac{1}{\sqrt{n_L}}\left\{\left\|\alpha_i^{(L)}(t)\right\|_{p^{in}} + c\left\|\boldsymbol{\Theta}_{\infty}^{(\boldsymbol{L})}\right\|_{op}\underbrace{\left\|\overbrace{\dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(t)\right)}^{\text{Lipschitz}}\right\|_{\infty}}_{|\dot{\sigma}|\leq c}\left\|W_{i,\bullet}^{(L)}\right\|_{L^2}\right\}\|d_t\|_{p^{in}}$$

$$\leq \frac{1}{\sqrt{n_L}}\left\{\left\|\alpha_i^{(L)}(t)\right\|_{p^{in}} + c^2\left\|\boldsymbol{\Theta}_{\infty}^{(\boldsymbol{L})}\right\|_{op}\left\|W_{i,\bullet}^{(L)}\right\|_{L^2}\right\}\|d_t\|_{p^{in}} \tag{III.2.7}$$

Now, with an easier example, notice two important aspects:

1. for three variables such that $z \geq x + y$ and a constant K a trivial inequality is:
$$(Kx + y) \leq \max\{K, 1\}z$$
2. in our case:
$$\left\|W_{i,\bullet}^{(L)}(t)\right\|_{L^2} + \left\|\alpha_i^{(L)}(t)\right\|_{p^{in}} = \left\|W_{i,\bullet}^{(L)}(t)\right\|_{L^2} \pm \left\|W_{i,\bullet}^{(L)}(0)\right\|_{L^2} + \|actL_i(t)\|_{p^{in}} \pm \left\|\alpha_i^{(L)}(0)\right\|_{p^{in}}$$
which by a trivial application of the reverse triangle inequality is less than:
$$\left\|W_{i,\bullet}^{(L)}(0)\right\|_{L^2} + \left\|W_{i,\bullet}^{(L)}(t) - W_{i,\bullet}^{(L)}(0)\right\| L^2 + \left\|\alpha_i^{(L)}(0)\right\|_{p^{in}} + \underbrace{\left\|\alpha_i^{(L)}(t) - \alpha_i^{(L)}(0)\right\|_{p^{in}}}_{=\sigma(\widetilde{\alpha}_i^{(L)}(t)) - \sigma(\widetilde{\alpha}_i^{(L)}(0))}$$
it follows by definition of Lipschitzness that this last value is also less than:
$$\left\|W_{i,\bullet}^{(L)}(0)\right\|_{L^2} + \left\|W_{i,\bullet}^{(L)}(t) - W_{i,\bullet}^{(L)}(0)\right\| L^2 + \left\|\alpha_i^{(L)}(0)\right\|_{p^{in}} + c\left\|\widetilde{\alpha}_i^{(L)}(t)) - \widetilde{\alpha}_i^{(L)}(0)\right\|_{p^{in}} = A(t)$$

Using the ideas of #1, #2 we could let:

$$K = c^2\left\|\boldsymbol{\Theta}_{\infty}^{(\boldsymbol{L})}\right\|_{op}, \quad x + y = \left\|W_{i,\bullet}^{(L)}(t)\right\|_{L^2} + \left\|\alpha_i^{(L)}(t)\right\|_{p^{in}} \leq A(t) = z$$

to continue the inequality chain that ended at Equation III.2.7 and derive a bound:

$$\partial_t A(t) \leq \frac{1}{\sqrt{n_L}}\max\left\{c^2\left\|\boldsymbol{\Theta}_{\infty}^{(\boldsymbol{L})}\right\|_{op}, 1\right\} A(t)\,\|d_t\|_{p^{in}}$$

The purpose of this big calculation was deriving a classical form of the derivative of an object as a bound of itself. This self-bound is the hypothesis needed for the classical Gronwall's Lemma (Lem. A.2.14), from which we derive a direct bound on $A(t)$:

$$A(t) \leq A(0)\exp\left\{\frac{c^2\max\left\{\left\|\boldsymbol{\Theta}_{\infty}^{(\boldsymbol{L})}{}_{op}\right\|, 1\right\}}{\sqrt{n_L}}\int_0^t \|d_s\|_{p^{in}}\,ds\right\} \tag{III.2.8}$$

where we have that:

- the red term is constant during training, and has an explicit form given by the inductive assumption of 🚗
- the blue term is bounded given the discussion of 🏛🦩
- by arguments similar to those of #2 in the list above we can define the following bounds for weights and activations:

$$\left\|\widetilde{\alpha}_i^{(L)}(t) - \widetilde{\alpha}_i^{(L)}(0)\right\|_{p^{in}} \leq \frac{1}{c}(A(t) - A(0))$$
$$\left\|W_{i,\bullet}^{(L)}(t) - W_{i,\bullet}^{(L)}(0)\right\|_{L^2} \leq A(t) - A(0)$$

  the flow of both norms in time is, by the exponential bound of Equation III.2.8, convergent to zero at rate $\mathcal{O}\left(\frac{1}{\sqrt{n_L}}\right)$. This convergence to nullity is at the sequential limit $n_1 \to \infty, \ldots, n_{L-1} \to \infty$.

(🏛🦩 subpoint, final discussion) we briefly recap what we have concluded to perform the last computations:

- we are in the inductive step
- the dynamics up to the L^{th} layer follow the differential equation claimed since d'_t is stochastically bounded
- at the last layer, the variations of weights and preactivations are bounded by a uniform $n_L^{-\frac{1}{2}}$ rate.

we are now ready to inspect the last layer update on the network function. For what concerns the bias, the contribution is always $\beta\delta_{jj'}$. Connection weights instead have as impact:

$$\partial_{W_{ij}^{(L)}} f_{\theta,j'}(x) = \frac{1}{\sqrt{n_L}}\alpha_i^{(L)}(x;\theta)\delta_{jj'} = \frac{1}{\sqrt{n_L}}\sigma\left(\underbrace{\widetilde{\alpha}_i^{(L)}(x;\theta)}_{\in\mathcal{O}\left(n_L^{-\frac{1}{2}}\right)}\right)\delta_{jj'}$$

which in time, by the Lipschitzness of σ and the already found bound on preactivations, suggests that the bound is again $\mathcal{O}\left(n_L^{-\frac{1}{2}}\right)$. In the NTK, this translates in the weight contribution having a rate:

$$\overbrace{\sum_{j'}^{n_L}\sum_{j}^{n_L}}^{\in\mathcal{O}(n_L^2)} \overbrace{\underbrace{\partial_{W_{ij}^{(L)}} f_{\theta,j'}(x)}_{\in\mathcal{O}\left(n_L^{-\frac{1}{2}}\right)} \otimes \underbrace{\partial_{W_{ij}^{(L)}} f_{\theta,j''}(x')}_{\in\mathcal{O}\left(n_L^{-\frac{1}{2}}\right)}}^{\in\mathcal{O}\left(n_L^{-\frac{3}{2}}\right)} \in \mathcal{O}\left(n_L^{-\frac{1}{2}}\right)$$

we are now missing only the contribution to the $L+1$-level NTK of the lower parameters. These are identified in the differential equation as:

$$\partial_{\widetilde{\theta}_k} f_{\theta,j}(x) = \frac{1}{\sqrt{n_L}}\sum_{i=1}^{n_L}\partial_{\widetilde{\theta}_k}\widetilde{\alpha}_i^{(L)}(x;\theta)\dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right)W_{ij}^{(L)}$$

The NTK contribution, specifically to $\boldsymbol{\Theta}^{(L+1)}_{jj'}$ is at finite neurons:

$$\frac{1}{n_L}\sum_{i,i'=1}^{n_L} \underbrace{\boldsymbol{\Theta}^{(L)}_{ii'}(x,x')}_{\sum_{k,k'}\partial_{\widetilde{\theta}_k}\widetilde{\alpha}^{(L)}_i(x;\theta)\otimes\partial_{\widetilde{\theta}_{k'}}\widetilde{\alpha}^{(L)}_{i'}(x;\theta)} \dot{\sigma}\left(\widetilde{\alpha}^{(L)}_i(x;\theta)\right) W^{(L)}_{ij}\dot{\sigma}\left(\widetilde{\alpha}^{(L)}_{i'}(x';\theta)\right) W^{(L)}_{i'j'}$$

The induction hypothesis of  states that at the sequential limit it holds that:

$$\lim_{n_L\to\infty}\cdots\lim_{n_1\to\infty}\boldsymbol{\Theta}^{(L)}_{ii'}(x,x') = \boldsymbol{\Theta}^{(L)}_{\infty}(x,x')$$

a result that we plug in the above equation. It follows by the bound on entries of the last layer weights found in III.2.6 that they have rate $\mathcal{O}\left(n^{-\frac{1}{2}}\right)$. What is missing is proving that this same rate is respected by the derivative of the sigmoid of the preactivations. To do so, we use the hypothesis of bounded second derivative, to conclude that:

$$\partial_t\left(\dot{\sigma}\left(\widetilde{\alpha}^{(L)}_i(x;\theta(t))\right)\right) \in \mathcal{O}\left(\underbrace{\partial_t\widetilde{\alpha}^{(L)}_i(x;\theta(t))}_{\in\mathcal{O}\left(n^{-\frac{1}{2}}\right)}\right) \in \mathcal{O}\left(n^{-\frac{1}{2}}\right)$$

using the $A(t)$ bound on preactivations. We have found **independently** of time a notion of convergence of the NTK to the starting NTK derived in Theorem III.1.4 at the divergent limit. The proof is complete by choosing n_L large enough as to capture the dynamics across all t. □

III.3 Dynamics Convergence

Recalling Observation I.3.13, a positive definite kernel is sufficient for convergence to a critical point. Such point is also optimal if the functional cost is **reasonably behaved**. To do so, we first introduce two previous results that will be needed.

Remark. *When referring to $\mathbb{S}^{d-2}$ we mean the unit sphere of a space $\mathbb{R}^{d-1}$. Namely:*

$$\mathbb{S}^{d-2} \coloneqq \left\{\theta\in\mathbb{R}^{d-1}\ ;\ |\theta|=1\right\}$$

Lemma III.3.1 (Dual via Hermite polynomials [DFS17](Lemma 12(a), Supplementary Material))**.** *Let $\widehat{\mu}:[-1,1]\to\mathbb{R}$ be the dual of a Lipschitz function $\mu:\mathbb{R}\to\mathbb{R}$ defined by:*

$$\widehat{\mu}(\rho) = \mathbb{E}_{(X,Y)}\left[\mu(X)\mu(Y)\right] \quad (X,Y)\sim\mathcal{GP}(0,\Sigma) \quad \Sigma = \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}$$

If μ is expanded in Hermite polynomials[6] as:

$$\mu = \sum_{i=0}^{\infty} a_i h_i$$

then its dual is expressed as:

$$\widehat{\mu}(\rho) = \sum_{i=0}^{\infty} a_i^2\rho^i$$

[6] We do not explain much about these objects. Take them as decompositions of functions via polynomials with well studied properties. A reference for those that wish to inspect the topic further is [Old+09](Chap. 24)

Theorem III.3.2 (An adaptation of [Gne13](Theorem 1(b))). *For a function:*

$$f : [-1, 1] \to \mathbb{R} \quad f(\rho) = \sum_{n=0}^{\infty} b_n \rho^n$$

the kernel:

$$\boldsymbol{K_f^{(n_0)}} : \mathbb{S}^{n_0-1} \times \mathbb{S}^{n_0-1} \to \mathbb{R} \quad \boldsymbol{K_f^{(n_0)}}(x, x') = f(x^T x')$$

is positive definite in the sense of Definition I.3.6 if and only if $b_n \geq 0$ for:

- *infinitely many even integers n*
- *infinitely many odd integers n*

where both conditions must hold simultaneously.

Our NTK is p.d. if the span of:

$$\partial_{\theta_p} F^{(L)} \quad p = 1, \dots, P$$

is dense in $\mathcal{F}$ with respect to $\|\cdot\|_{p^{in}}$ (Def. A.2.9) as $n_1 \to \infty, \dots, n_L \to \infty$ sequentially [JGH20]. This idea is intuitively justified by the fact that the objects that compose the rows of the NTK are able to arbitrarily approximate the space of functions, thus making their inner product strictly positive. For more context, refer to [SC04].

Natural Setting Taking the span of the last preactivations $\widetilde{\alpha}^{(L)}$, appearing in $\partial_{\theta_p} F^{(L)}$, and identified by $W^{(L-1)}$, the $\mathcal{F}$-p^{in} density of those objects is pretty much part of the foundational results on Neural Networks. We are indeed asserting that it is possible to simulate with arbitrary precision functions belonging to a space via the parameters of a neural network. These density results include different kinds of assumptions on non-linearities. As a starting point, one could refer to [HSW89; Les+93]. The term *natural setting* is thus used to stress the fact that these are well known properties of Neural Networks, approachable with the classic parametric formulation.

As a matter of fact, we need slightly more in our case, as we consider $\partial_{\theta_p} F^{(L)}$ in its entirety to show that the NTK is positive definite. Below, a result is reported for instances of $\mathcal{D}$ supported on the unit sphere.

Proposition III.3.3 (Sphere data NTK is positive definite). *Consider a non-linearity σ which is:*

- *Lipschitz*
- *non polynomial*

then, for $L \geq 2$ the restriction to the sphere $\mathbb{S}^{n_0-1}$ of the limiting NTK $\boldsymbol{\Theta_\infty^{(L)}}$ derived in Theorems III.1.4, III.2.3 is positive definite in the sense of Definition I.3.6.

Remark. *The requirements could be summarized as follows.*

- *a specific form on the non-linearity*
- *at least one hidden layer*
- *evaluating the NTK only on values lying on the unit sphere.*

Proof. (**strategy**) we perform a recursive decomposition of $\boldsymbol{\Theta^{(L)}}$, which allows to check if it is positive definite via the kernels of the activations. A sufficient condition for all of them to

be positive definite is that $\boldsymbol{\Theta^{(2)}}$ is.

(🪝 **one step decomposition)** For any $L \geq 1$ we have by Theorem III.1.4:

$$\boldsymbol{\Theta^{(L+1)}} = \dot{\Sigma}^{(L)}\boldsymbol{\Theta^{(L)}} + \Sigma^{(L+1)}$$

A sufficient condition for the $(L+1)^{th}$ NTK to be positive definite is that all the three terms are positive definite since the product of p.d. kernels is p.d., as well as the sum[7]. We then find that:

- $\dot{\Sigma}^{(L+1)} = \mathbb{E}_{f\sim\mathcal{N}(0,\Sigma^{(L)})}\left[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))\right]$ is p.d. by the recursive argument
- $\boldsymbol{\Theta^{(L)}}$ is p.d. by the recursive search

So, we just need to show that $\Sigma^{(L+1)}$ is p.d.

(🚗 **sufficiency on $\Sigma^{(L+1)}$ p.d.ness)** Using Proposition III.1.2 we write:

$$\Sigma^{(L+1)}(x,x') = \mathbb{E}_{f\sim\mathcal{N}(0,\Sigma^{(L)})}\left[\sigma(f(x))^T\sigma(f(x'))\right] + \beta^2$$

so that for any $\vec{\boldsymbol{c}} \in \mathbb{R}^d$ and distinct $\{x_i\}_{i=1}^d \in \mathbb{R}^{n_0}$ we have:

$$\begin{aligned}
\sum_{i,j=1}^d c_i c_j \Sigma^{(L+1)}(x_i,x_j) &= \sum_{i,j}^d c_i c_j \left(\mathbb{E}_{f\sim\mathcal{N}(0,\Sigma^{(L)})}\left[\sigma(f(x_i))^T\sigma(f(x_j))\right] + \beta^2\right) \\
&= \sum_{i,j=1}^d c_i c_j \left(\mathbb{E}\left[\sigma(f(x_i))^T\sigma(f(x_j))\right] + \beta^2\right) \qquad \text{simplified notation} \\
&= \sum_{i,j=1}^d c_i c_j \mathbb{E}\left[\sigma(f(x_i))^T\sigma(f(x_j))\right] + \sum_{i,j}^d c_i c_j \beta^2 \\
&\left[\sum_{i,j} q_i q_j = \sum_i q_i^2 + 2\sum_{i<j} q_i q_j = \left(\sum_i q_i\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_i c_i \sigma(f(x_i))\right)^2\right] + \beta^2\left(\sum_i c_i\right)^2
\end{aligned}$$

If such quantity is null, then the kernel is only p.s.d. A necessary condition for this is that the first term is null for some $\vec{\boldsymbol{c}} \neq \boldsymbol{0}$. In this context, with a p.d. kernel $K^{(L)}$, which guarantees that the sampled collection $(f(x_i))_{i=1}^d$ is non degenerate, by the non polynomial form of σ, forces all coefficients to be null if the first term is null, namely $\vec{\boldsymbol{c}} = \boldsymbol{0}$. Basically, the non-polynomial form of σ makes the necessary condition of the first term to vanish verify only for $\vec{\boldsymbol{c}} = \boldsymbol{0}$ which is not allowed. This shows that $\Sigma^{(L)}$ positive definite is a sufficient condition for $\Sigma^{(L+1)}$ to be positive definite.

(🐦‍⬛ **inductive argument)** if $\boldsymbol{\Sigma^{(2)}}$ is p.d., then all $\Sigma^{(L)}$ for $L \geq 2$ are, thanks to 🚗. This would also ensure that $\boldsymbol{\Theta}^{(L+1)}$ is p.d. by the discussion in 🪝.

(🏠 **focus on $\Sigma^{(2)}$)** notice that we restricted further the argument to a sphere, so we need to

[7]The implications are almost immediate, use Def. I.3.4 and Def. I.3.6 which are a description of positive definite kernels. Clearly, sums and products obey the same laws

check:

$$\Sigma^{(2)} = \mathbb{E}_{(X,Y)\sim\mathcal{N}(0,\widetilde{\Sigma})}\left[\sigma(X)\sigma(Y)\right] + \beta^2 \quad \widetilde{\Sigma} = \begin{bmatrix} \frac{1}{n_0} + \beta^2 & \frac{1}{n_0}x^T x' + \beta^2 \\ \frac{1}{n_0}x^T x + \beta^2 & \frac{1}{n_0} + \beta^2 \end{bmatrix}$$

with $\widetilde{\Sigma}$ being the covariance of the first layer adapted to x, x'. If we choose as function:

$$\mu : \mathbb{R} \to \mathbb{R} \quad x \to \mu(x) = \sigma\left(x\sqrt{\frac{1}{n_0} + \beta^2}\right)$$

we can say that its Dual in the sense of Lemma III.2.2 is constructed as a rescaled version of the Gaussian sampled (X, Y) of the previous covariance function. We are indeed collecting a factor equal to the diagonal inside the argument of μ, and will get that the new covariance from which we sample is:

$$\widetilde{\Sigma}_{rescaled} = \begin{bmatrix} 1 & \frac{\frac{1}{n_0}x^T x' + \beta^2}{\frac{1}{n_0} + \beta^2} \\ \frac{\frac{1}{n_0}x^T x' + \beta^2}{\frac{1}{n_0} + \beta^2} & 1 \end{bmatrix} \quad \rho = \frac{\frac{1}{n_0}x^T x' + \beta^2}{\frac{1}{n_0} + \beta^2} = \frac{n_0\beta^2 + x^T x'}{n_0\beta^2 + 1}$$

which in turn means:

$$\begin{aligned} \Sigma^{(2)}(x, x') &= \mathbb{E}\left[\sigma(X)\sigma(Y)\right] + \beta^2 \\ &= \mathbb{E}_{(X,Y)}\left[\mu(X)\mu(Y)\right] + \beta^2 && \text{rescaled} \\ &= \mathbb{E}_{(X,Y)}\left[\sigma\left(X\sqrt{\frac{1}{n_0} + \beta^2}\right)\sigma\left(Y\sqrt{\frac{1}{n_0} + \beta^2}\right)\right] + \beta^2 \\ &= \widehat{\mu}(\rho) + \beta^2 \\ &= \widehat{\mu}\left(\frac{n_0\beta^2 + x^T x'}{n_0\beta^2 + 1}\right) + \beta^2 \end{aligned}$$

which, by the result of Lemma III.3.1 has respective Hermite expansions:

$$\mu = \sum_{i=0}^{\infty} a_i h_i \quad \widehat{\mu}(\rho) = \sum_{i=0} a_i^2 \rho^i$$

By nonpolyomial assumption on σ (and equivalently μ) the polynomials used to approximate it have a non-null coefficient a_i at infinite locations.
Roughly, if σ was a polynomial then at some point the sum would have stopped, since polynomials can be expressed as a finite sum of polynomials. Contrarily, a nonpolynomial function is only approximated by a sum of polynomials.

(finalization) an application of the last form of $\Sigma^{(2)}$ derived and the discussion on the Hermite expansion suggests that:

$$\Sigma^{(2)}(x, x') = \nu(x^T x) = \beta^2 + \sum_{i=0}^{\infty} a_i^2 \left(\frac{n_0\beta^2 + x^T x'}{n_0\beta^2 + 1}\right)^i$$

by the a_i coefficients being infinitely non-null, there are infinitely many odd non-null and infinitely many even non-null coefficients. By Theorem III.3.2, we conclude that $\Sigma^{(2)}$ is positive definite. By , $\Sigma^{(L)}$ is p.d. for all $L \geq 2$, by , $\boldsymbol{\Theta^{(L+1)}}$ is p.d. for all $L \geq 1$. □

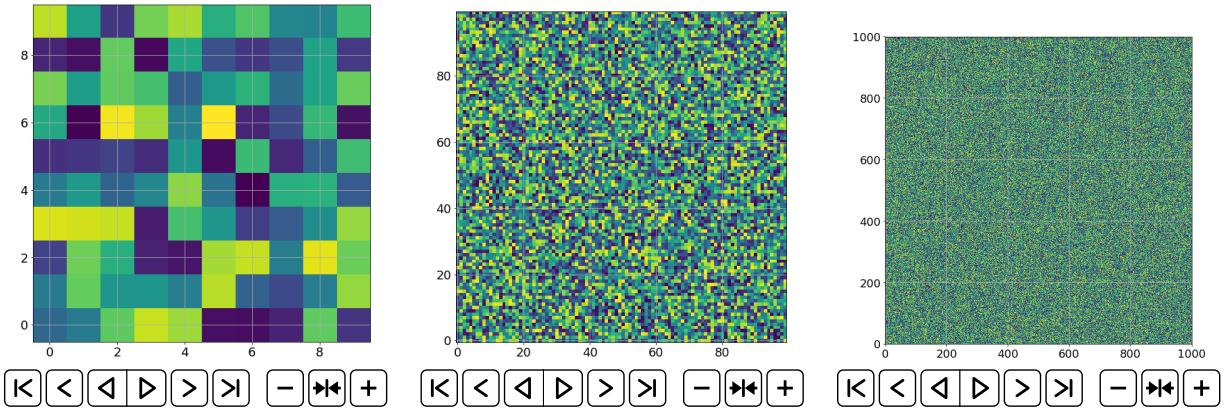


Figure 4: Weight matrix dynamics, small medium and large size. Source [Vad19]

Observation III.3.4 (Partial converse)**.** *An argument like the proof of the above proposition for polynomial non-linearities concludes that there exists n_0 such that the kernel $\mathbf{\Theta^{(2)}}$ is not positive definite on $\mathbb{S}^{n_0-1}$.*

IV Phenomenology

With the properties of Section III available, discussion on three applications is proposed. The first is an interpretation of learning processes that stay close to the starting configuration in terms of the newly found NTK. The second is a justification of early stopping heuristics with continuous valued data and the classic square loss. Last but not least, we quickly provide a Bayesian interpretation of the kernel linearization of the problem.

IV.1 Linearized dynamics as a competitor of Neural Networks

Empirically, at large hidden neurons sizes, the θ matrix is *almost* static. Optimization of the parameters, though changing, is roughly constant, in the sense that there are similar patterns across trajectories. This phenomenon is called **lazy training**.

Example IV.1.1 (Lazy training in practice)**.** *The dynamics at increasing network size for $L = 2$ are visualized in Figure 4. Despite being GIFs, changes are noticeable only in the smallest one.*

This is not always the case, it is a **divergent** behavior. It is also an empirical observation up to now. It does not mean the model will take longer to train,but just that convergence in the parameter space is at a closer point. The updates are local, in a ball around $\theta(0)$. We look at a special region where all local minima are global, without tweaking the optimization landscape. The limitation is that we identify this region, but we might not care about it[8].
Based on this intuition, we could Taylor approximate the update.

$$f_\theta(x) \approx f_{\theta(0)}(x) + \partial_\theta f_{\theta(0)}(x)^T(\theta - \theta(0)) + h.o.t.$$

where:

[8] e.g. what if we care about generalization?

- *h.o.t.* means *higher order terms*
- the function is affine in θ or in $\Delta(\theta) = \theta - \theta(0)$

Is this model linear in θ? Yes
Is this model linear in x? No, the dependence comes from ∂_θ, and it is potentially non-linear by the non-linear activations.

Assumption IV.1.2 (Intercept)**.** *Notice $f_{\theta(0)}(x)$ is fixed in θ and we could just put $f_{\theta(0)}(x) = 0 \forall x$. This could be done with two networks summing up and subtracting up. Think for example of two symmetric ANNs:*

$$\sum -b_i \sigma(\vec{\boldsymbol{a}}_i^T x) + \sum b_i \sigma(\vec{\boldsymbol{a}}_i^T x) = 0$$

For this reason, from now on we assume it to be zero.

Definition IV.1.3 (Linearized Model g_θ)**.** *With Assumption IV.1.2, define:*

$$g_\theta(x) \coloneqq f_{\theta(0)}(x) + \left\langle \partial_\theta f_{\theta(0)}(x), \theta - \theta(0) \right\rangle$$

We could set $\partial_\theta f_{\theta(0)}(x) = \varphi(x)$, and the expansion looks like gradient descent with the kernel method of Subsection I.3. Recall that we evaluated similarities between inputs over a higher dimensional space:

$$\mathbf{K}(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$$

We can then interpret the expression $\widehat{y}' = \widehat{y} - f_{\theta(0)}(x) = \left\langle \partial_\theta f_{\theta(0)}(x), \Delta\theta \right\rangle$ as a feature map with parameter $\Delta\theta$. By Assumption IV.1.2, $\widehat{y}' = \widehat{y}$ and:

$$\mathbf{K}(x, x') = \left\langle \varphi(x), \varphi(x') \right\rangle = \left\langle \partial_\theta f_{\theta(0)}(x), \partial_\theta f_{\theta(0)}(x') \right\rangle$$

Observation IV.1.4 (Linearized model)**.** *Optimizing g_θ is easy since:*

- *it is a **linear** model*
- *we have a notion of Kernel for the higher dimensional space*
- *the functional cost is **convex** in such space*
- *the update $\left\langle \partial_\theta f_{\theta(0)}, \Delta\theta \right\rangle$ is the **gradient descent**/gradient flow Equation*

Observation IV.1.5 (Why does this make sense?)**.** *By the results of Section III, expanded also without the sequential limit requirement [Aro+19], the NTK at the start is constant and deterministic. The lazy training phenomenon is then justified by the stability of kernel methods, which are not affected by the gradient flow updates (recover the discussion of pros and cons of Subsection I.3). At the infinite-width limit, we are effectively simulating a linear nonparametric model, as explained below.*

Consider the empirical loss:

$$\mathscr{L}(f_\theta; \vec{\boldsymbol{y}}, \mathbf{X}) = \mathscr{L}(f_\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i)$$

with $\mathscr{L}(g_\theta)$ specified accordingly[9]. If we use g_θ instead of f_θ, we have a kernel method, a linear model on top of the features. In particular, when $\theta \approx \theta(0)$ the loss is an approximation as well, element-wise:

$$\mathcal{L}(f_\theta(x), y) \approx \mathcal{L}(g_\theta(x), y)$$

[9] If we do not use a subscript, then we implicitly refer to $\mathscr{L}(f_\theta)$

However, the latter is convex by $\mathcal{L}$ being convex and g linear. The question becomes how valid is this approximation. Under the results we showed in Section III, the approximation is valid at the infinite-width limit.

Under the NTK regime, the Neural Network is comparable to its linearized version, losing the advantage of non-linear models discussed across our document. The result is at divergent size, but is nevertheless groundbreaking in terms of understanding these learning structures.

Observation IV.1.6 (Warning)**.** *A tempted reader might then conclude that Neural Networks are ultimately useless as they are just as good as linear models, but there are many reasons why this is not accurate, such as:*

- *ANNs as in Definition I.2.2 are not representative of the whole spectrum of NNs*[10]
- *though not much restrictive, we are adding the structure of Assumption I.2.7*
- *the results are at the infinite-width limit*

Additional arguments are added in Section VI to close the document.

The motivation for the acronym NTK can be easily justified by the linearization result.

- **N**eural since it is used for neural networks
- **T**angent since is is evaluated at the tangent plane at θ_0
- **K**ernel since it is a kernel method

IV.2 Quadratic Cost ANN regression

In practical terms it is often the case that $\mathscr{Y}$ is an uncountable space such as the whole $\mathbb{R}^{n_L}$. While this is not required in principle, it is the easiest possible setting in which the aim is inferring continuous n_L-dimensional labels. A classical choice for the cost of an ANN is then **least squares regression cost**. Provided that we have a goal function f^* to estimate, under the empirical distribution as per Assumption I.2.6, it will have form:

$$C(f) = \frac{1}{2}\left\|f - f^*\right\|_{p^{in}}^2 = \frac{1}{2}\mathbb{E}_{x\sim p^{in}}\left[\left\|f(x) - f(x^*)\right\|^2\right] \tag{IV.2.1}$$

where the norm inside the expectation is just the euclidean norm to make the result map to $\mathbb{R}$. Additionally, provided with a dataset $\mathcal{D}$ the empirical norm used to sample makes it clear that $f^*(x_i) = y_i \in \mathscr{Y} \forall i$.
Under Assumptions I.2.7 for the nonlinearities of an ANN as in Definition I.2.2, both the requirements of Theorems III.1.4, III.2.3 are satisfied.

Proposition IV.2.2 (Regression training direction)**.** *The training direction $d|_f$ with a cost functional as in Equation IV.2.1 is:*

$$\frac{\delta C}{\delta f} = d|_f = f - f^*$$

[10] on a side note, [Aro+19] finds the NTK of Convolutional Neural Networks.

Proof. **(📐 strategy)** we use the definition of functional derivative for $\phi \in \mathcal{F}$.

(🪝 preliminary expression) for clarity, we do the intermediate step:

$$\begin{aligned} C(f+\epsilon\phi) &= \frac{1}{2}\mathbb{E}_{x\sim p^{in}}\left[\|(f(x)+\epsilon\phi)-f^*(x)\|^2\right] \\ &= \frac{1}{2}\mathbb{E}_{x\sim p^{in}}\left[\|f(x)-f^*(x)\|^2 + 2\epsilon\,\langle f(x)-f^*(x),\phi\rangle + \epsilon^2\,\|\phi(x)\|^2\right] \end{aligned}$$

where the norms and the inner product are Euclidean.

(🚗 functional derivative) using Definition A.2.3 we conclude that:

$$\begin{aligned} \lim_{\epsilon\downarrow 0}\frac{C(f+\epsilon\phi)-C(f)}{\epsilon} &= \lim_{\epsilon\downarrow 0}\left(\mathbb{E}_{x\sim p^{in}}\left[\langle f(x)-f^*(x),\phi\rangle\right] + \frac{\epsilon}{2}\mathbb{E}_{x\sim p^{in}}\left[\|\phi(x)\|^2\right]\right) \\ &= \mathbb{E}_{x\sim p^{in}}\left[\langle f(x)-f^*(x),\phi\rangle\right] \\ &= \langle f(x)-f^*(x),\phi\rangle_{p^{in}} \end{aligned}$$

and the functional derivative is as claimed. □

With this premise, the $\|d(f)\|_{p^{in}}$ decreases along time, and the integral $\int_0^T \|d(f)\|_{p^{in}}\,dt$ is bounded for any T.

We are now able to inspect the learning dynamics in terms of the kernel gradient (Def. I.3.11):

$$\partial_t f_t = \Phi_{\mathbf{K}}\left(\langle f-f^*,\cdot\rangle_{p^{in}}\right)$$

in particular for the kernel $\mathbf{K}=\boldsymbol{\Theta}_{\infty}^{(L)}$ we derived in Theorems III.1.4, III.2.3.
Though apparently difficult, it is one of the easiest forms an ODE can take. In order to solve it, we consider a given starting point f_0 and preconstruct a map:

$$\Pi : \mathcal{F}\to\mathcal{F} \quad f\to\Phi_{\mathbf{K}}\left(\langle f,\cdot\rangle_{p^{in}}\right) \tag{IV.2.3}$$

which for a finite dataset $\{x_1\dots,x_N\}$ is:

$$\Pi(f)_k : \mathscr{X}\to\mathscr{Y} \quad x\to\Pi(f)_k(x)=\frac{1}{N}\sum_{i=1}^{N}\sum_{k'=1}^{n_L} f_{k'}(x_i)\mathbf{K}_{k,k'}(x_i,x) \tag{IV.2.4}$$

Proposition IV.2.5 (Regression dynamics closed form)**.** *With a cost functional as in Equation IV.2.1 imposing the kernel gradient differential equation:*

$$\partial_t f_t = \Phi_{\boldsymbol{K}}\left(\langle f-f^*,\cdot\rangle_{p^{in}}\right)$$

which is at the infinite-width realization function f_t, we find that:

$$f_t(x) = f^*(x) + e^{-t\Pi}(f_0-f^*)$$

Proof. **(📐 strategy)** we check that the claim is correct, rather than deriving it.

(🪝 exponential map) notice that:

$$e^{-t\Pi}(f)=\sum_{k=0}^{\infty}\frac{(-t)^k}{k!}\Pi^k(f) \quad f\in\mathcal{F}$$

which is the exponential of a map in the sense of Definition A.2.10.

(↰ solution derivative) if we consider the claimed form, deriving with respect to time:

$$\begin{aligned}
\partial_t f_t &= \partial_t \left(f^*(x) + e^{-t\Pi}(f_0 - f^*) \right) \\
&= \partial_t \left(e^{-t\Pi}(f_0 - f^*) \right) \\
&= \partial_t \left(\sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \Pi^k (f_0 - f^*) \right) && \text{↰} \\
&= \left(\sum_{k=0}^{\infty} \partial_t \frac{(-t)^k}{k!} \Pi^k (f_0 - f^*) \right) \\
&= \sum_{k=1}^{\infty} \frac{(-1)k(-t)^{k-1}}{k!} \Pi^k (f_0 - f^*) && \text{use } k' = k-1 \\
&= -\sum_{k'=0}^{\infty} \frac{(-t)^{k'}}{(k')!} \Pi^{k'+1} (f_0 - f^*)
\end{aligned}$$

where the index k' is a dummy index and could be exchanged with k for clarity.

(🐟 on the form of Π) if the claim is correct, we also have that:

$$f^* - f = -e^{-t\Pi}(f_0 - f^*) = -\sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \Pi^k (f_0 - f^*)$$

which, if evaluated via the map Π, by its linearity (see how it was constructed in Equation IV.2.3):

$$\begin{aligned}
\Phi_{\mathbf{K}} \left(\langle f^* - f, \cdot \rangle_{p^{in}} \right) &= \Pi(f^* - f) \\
&= \Pi \left(-\sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \Pi^k (f_0 - f^*) \right) && \text{above argument} \\
&= -\sum_{k=0}^{\infty} \frac{(-t)^k}{(k)!} \Pi^{k+1} (f_0 - f^*) && \text{linearity} \\
&= \partial_t f_t && \text{↰ result}
\end{aligned}$$

(🐦‍⬛ boundary conditions) notice that we derived thiis result using $\partial_t f_t$ but for any constant $\partial_t(f_t + k) = \partial_t(f_t)$, so we have been slightly informal on the discussion about boundary conditions. It is however sufficient to check that for $t = 0$ we must have $f_t = f_0$, so the constant term is f^*. □

We eventually state that:

$$f_t = f^* + e^{-t\Pi}(f_0 - f^*) \tag{IV.2.6}$$

where it is possible to notice that convergence to f^* is exponentially in time.
The seemingly strange exponential of Π is intuitively expressed in the Taylor sum sense (for more context, see Def. A.2.10 *et seq.*) as:

$$e^{-t\Pi} = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \Pi^k$$

Assume now the kernel is positive definite by being constructed over spherical data, which is the result of Proposition III.3.3. This means that all its eigenvalues are all bounded away from zero. Combined with the symmetry notion explained in Definition I.3.6, the matrix $\mathbf{K}$ is diagonalizable. Further, by the linearity of $e^{-t\Pi}$ which is intuitive, given that Π diagonalizable as well by eigenfunction-eigenvalue pairs (Def. A.2.11) of the form $(\lambda_i, f^{(i)})$, it will be that the eigenfunctions and eigenvalues of $e^{-t\Pi}$ are, respectively, a rescaled version and the same. Namely:

$$\left\{\left(e^{-\lambda_i}, f^{(i)}\right)\right\}_{i=1}^{Nn_L} \quad \text{or less in number}$$

where the $f^{(i)}$ are positive.
We deduce from easier cases that:

- The function collection $\{f^{(1)}, \dots, f^{(Nn_L)}\}$ is that of the kernel principal components of the data with respect to $\mathbf{K}$ [SSM98; SC04]
- λ_i is a variance notion of the i^{th} eigenspace

Given that $f^* - f_0$ is in $\mathcal{F}$, we could equivalently express their difference as a sum of differences along the orthogonal Π-eigenspaces:

$$f^* - f_0 = \Delta_f^0 + \cdots + \Delta_f^{Nn_L} \quad \Delta_f^i \propto f^{(i)} \tag{IV.2.7}$$

where, with a slight abuse of notation and syntax we denote the null-space (also kernel, here the syntax overlap) as Δ_f^0, which has[11] *"eigenvalue"* $\lambda_0 = 0$.
‘ Eventually, we reformulate Equation IV.2.6 as:

$$\begin{aligned} f_t &= f^* + e^{-t\Pi}(f_0 - f^*) \\ &= f^* + \sum_{i=0}^{Nn_L} e^{-t\lambda_i}(\Delta_f^i) \\ &= f^* + \Delta_f^0 + \sum_{i=1}^{Nn_L} e^{-t\lambda_i}(\Delta_f^i) \end{aligned} \tag{IV.2.8}$$

Observation IV.2.9 (Early stopping in Neural Networks)**.** *The dynamics expressed as in Equation IV.2.8 give a nice interpretation in terms of early stopping. The decoupling induced by the spectral decomposition of the exponential matrix allows the dynamics to be separated along the eigenspaces of* Π*. Among all these* Nn_L *directions, the speed of convergence is different and governed by* λ_i*. Thus, the bigger the* λ_i*, the bigger the variation inside the eigenspace, the faster the convergence.*
In this context, early stopping comes into play as a naïve techinque to avoid generalization problems. Heuristically, eigenspaces are derived with respect to the sample $(x_1, \dots, x_N)$*, and are not necessarily representative of the entire population. The lower eigenvalue spaces loosely represent regions of this hypothesized model where the variation is low. To a low variation, assuming that the sample does not suffer from pathological issues, we associate noise. Early stopping stops before convergence, subject to some criteria, to avoid fitting the noise. In the derived eigendecomposition, we let the learning flow until* ***not all of the directions*** *have saturated. Clearly, those associated to a greater* λ_i *will have converged faster.*

[11] Recall that $\texttt{Ker}(T) = \{x \in \mathscr{X} \mid f(x) = 0 = 0x\}$. The null coefficient is not a valid eigenvalue but is paired with the null space regardless

Example IV.2.10 (Early stopping in One Layer Neural Networks)**.** *Consider a quadratic loss in the simple one hidden layer setting of $L = 2, n_L = 1$ from Subsection II.2. Mathematically:*

$$\mathscr{L}(\theta) = \frac{1}{2}\left\|\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right\|^2 \quad \vec{\boldsymbol{y}} \in \mathbb{R}^N, \widehat{\vec{\boldsymbol{y}}} \in \mathbb{R}^N \quad \widehat{\vec{\boldsymbol{y}}} = \begin{bmatrix} f_\theta(x_1) \\ \vdots \\ f_\theta(x_N) \end{bmatrix}$$

The gradient is:

$$\partial_\theta \mathscr{L}(\theta) = \left(\partial_\theta \widehat{\vec{\boldsymbol{y}}}\right)^T \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right)$$

The gradient flow becomes:

$$\partial_t \theta(t) = \frac{d\theta(t)}{dt} = -\left(\partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}}\right)^T \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right)$$

This shows the dynamics of θ in the parameter space across time. We could also look at the dynamics of the output in the parameter space:

$$\begin{aligned}
\frac{d\widehat{\vec{\boldsymbol{y}}}}{dt} &= \left(\partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}}\right)^T \frac{d\theta(t)}{dt} && \text{chain rule} \\
&= -\left(\partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}}\right)^T \partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}} \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right) && \text{above result} \\
&= -\left\|\partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}}\right\|^2 \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right) \\
&\approx -\boldsymbol{K}(\theta(0))(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}})
\end{aligned}$$

where the norm can be approximated by the NTK evaluated at θ_0 at training, which is a good representative of the dynamics of the kernel gradient by the results of Section III.
Now define $\vec{\boldsymbol{u}} = \widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}$ and see that:

$$\frac{d\vec{\boldsymbol{u}}}{dt} = \frac{d\widehat{\vec{\boldsymbol{y}}}}{dt} \approx \boldsymbol{K}(\theta(0)) \cdot \vec{\boldsymbol{u}} \overset{ODE}{\Longrightarrow} \vec{\boldsymbol{u}}(t) = \vec{\boldsymbol{u}}(0) e^{-\boldsymbol{K}(\theta(0))t}$$

If the NTK matrix becomes positive definite, the minimum eigenvalue is nonzero, and all of them are positive. Assuming that there are no null eigenvectors and no multiple eigenvalues, the eigendecomposition of $\boldsymbol{K}$ has form:

$$\boldsymbol{K}(\theta(0)) = \sum_{i=1}^N \lambda_i \vec{\boldsymbol{v}}_i \vec{\boldsymbol{v}}_i^T \quad 0 < \lambda_1 < \cdots < \lambda_N$$

and we eventually get:

$$\vec{\boldsymbol{u}}(t) = \vec{\boldsymbol{u}}(0) \prod_{i=1}^N e^{-t\lambda_i \vec{\boldsymbol{v}}_i \vec{\boldsymbol{v}}_i^T}$$

It is easy to conclude by noting that exponential convergence has rate $\min\{\lambda_i\} = \lambda_1$, and we recover the early stopping justification of this particular case for the orthogonal eigenspaces identified by $\{\vec{\boldsymbol{v}}_i\}_{i=1}^N$.

IV.3 Bayesian viewpoint

We now consider the idealistic setting in which all of our previous discussions are verified. This will allow us to perform a comment on the arising statistical model. Recollecting ideas:

1. we are at the infinite-width limit
2. the functional cost is the regression functional cost (Eqn. IV.2.1)
3. the kernel $\boldsymbol{K}$ is positive definite
4. the flow in time is not constrained, and $t \to \infty$, with the Π matrix diagonalizable (i.e. with all eigenvalues, the most favourable case)

From point #1 Proposition III.1.2 concludes that f_0 is Gaussian distributed. By the linearity of $e^{-t\Pi}$, in the derived dynamics of the cost #2, we conclude that f_t will be Gaussian for all iterations.
Thanks to #3, the Gram matrix (Def. A.2.12):

$$\widetilde{\boldsymbol{K}} = ((\boldsymbol{K}_{kk'}(x_i, x_j)))_{ik,jk'} \in \mathbb{R}^{Nn_L \times Nn_L}$$

is invertible (Prop. A.2.13).
The limit of the dynamics in #4 instead is fundamental to give a closed form for the entries of:

$$\begin{aligned}
\lim_{t\to\infty} f_t &= f_\infty && \text{Gaussian} \\
&= f^* + \Delta_f^0 && \text{all directions brought to zero} \\
&= f_0 + \sum_{i=1}^{Nn_L} \Delta_f^i && \text{Eqn. IV.2.7}
\end{aligned}$$

which is such that:

$$f_{\infty,k}(x) = \underbrace{\kappa_{x,k}^T \widetilde{\boldsymbol{K}}^{-1} y^*}_{blue} + \underbrace{\left(f_0(x) - \kappa_{x,k}^T \widetilde{\boldsymbol{K}}^{-1} y_0\right)}_{red} \quad \forall k \in 1, \dots, n_L, \forall x \in \mathbb{R}^{n_0}$$

where the three vectors we introduced are:

$$\begin{aligned}
\kappa_{x,k} &= (\boldsymbol{K}_{kk'}(x, x_i))_{i,k'} \in \mathbb{R}^{Nn_L} && \text{similarity notion} \\
y^* &= (f_k^*(x_i))_{i,k} \in \mathbb{R}^{Nn_L} && \text{best guess} \\
y_0 &= (f_{0,k}(x_i))_{i,k} \in \mathbb{R}^{Nn_L} && \text{first guess}
\end{aligned}$$

Notice that the index here is varying in its range so that all of them are vectors.

The *blue* is the **mean** and could be seen as the Maximum a Posteriori (MAP) with Gaussian priors on each of the functions, with the added constraint to be equal on the dataset to the target estimator. Namely :

$$f_k \sim \mathcal{N}\left(0, \boldsymbol{\Theta}_\infty^{(\boldsymbol{L})}\right) \quad f_k(x_i) = f_k^*(x_i) \quad \forall i \in \{1, \dots, N\}$$

The red term is a zero mean Gaussian with variance vanishing at the point in the dataset (namely, null in the dataset).

To understand this intuitively, recall Equation IV.2.4. Specifying the single entry k it reads:

$$\Pi(f)_k(x) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k'=1}^{n_L} f_{k'}(x_i) \mathbf{K}_{kk'}(x_i, x) = \frac{1}{N} \sum_{i=1}^{N} g_i(x) \quad g_i(\cdot) = f(x_i)\mathbf{K}(x_i, \cdot) \quad g_i : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$$

where the span of the g_i functions represents the range of Π, since we are linearly combining them. By this, conclude that the explainable part is a linear combination of those g_i functions, meaning:

$$\sum_{i=1}^{Nn_L} \Delta^i_{f_k}(x) = \kappa^T_{x,k}\vec{\boldsymbol{c}} \quad \vec{\boldsymbol{c}} \in \mathbb{R}^{Nn_L}$$

This combination logically minimizes the error of the orthogonal projection onto the space spanned by $\{g_i\}_{i=1}^N$, which is

$$\arg\min_{\alpha \in \mathbb{R}^{Nn_L}} \left\| f^*_k - f_{0,k} - \kappa^T_{x,k}\vec{\boldsymbol{c}} \right\|^2_{\mathbf{K}} \implies \vec{\boldsymbol{c}} = \widetilde{\boldsymbol{K}}^{-1}(f^* - f_0) \quad f^* = y^*$$

Observation IV.3.1 (Kernel Methods connection)**.** *Considering Kernel Ridge regression [SSM98] with the regularization coefficient $\lambda \to 0$, the estimators are equivalent.*

V Experiments

We present a selection of results almost completely from the main publication [JGH20]. However, readers who are interested in seeing more are invited to visit the PapersWithCode thread of NTK implementations, which is a nice collection.
Having derived an approximation via a Kernel of ANNs at the infinite-width limit, the two models will be compared against datasets. For practical purposes, the choice of the specifications is limited to:

- equal size hidden layers $n \coloneqq n_1 = \cdots = n_{L-1}$
- `ReLu` non linearity $\sigma(x) = \max\{0, x\}$
- $n_L = 1$ since by the authors results $n_L > 1$ is equal to training n_L independent networks with scalar $y \in \mathbb{R}$ outputs dy the decoupling of the $\otimes$ operation across the output dimensions
- the only parameter showing up in Equation I.2.9 is chose to be $\beta = 0.1$, which is experimentally in line with the comments of Observation I.2.10

In order to corroborate the analysis, the two important discussions about the peculiarities of the Kernel and the ties with Regression are inspected. The datasets $\mathcal{D}$ for the first two examples are synthetic, and created as to be supported on a sphere $\mathbb{S}^{n_0-1}$ where $n_0 = 2$. They are basically **points on a circle**. By construction, we are allowed to use Proposition III.3.3, to conclude that $\boldsymbol{\Theta}^{(L)}$ is positive definite and the dynamics converge.
This choice might look like an over-simplification, but is actually in line with the high-dimensional data regime, where centered data points have approximately the same norm. For an example, one could consider the n_0 dimensional Gaussian distribution, which for $n_0 \to \infty$ bounds the sampled norm as $\Omega(\sqrt{n_0})$. A source discussing this matter further is [Cha+12].

V.1 NTK convergence

We choose:

- $L = 4$
- $n \in \{500, 10000\}$ as size of the hidden layers for comparison

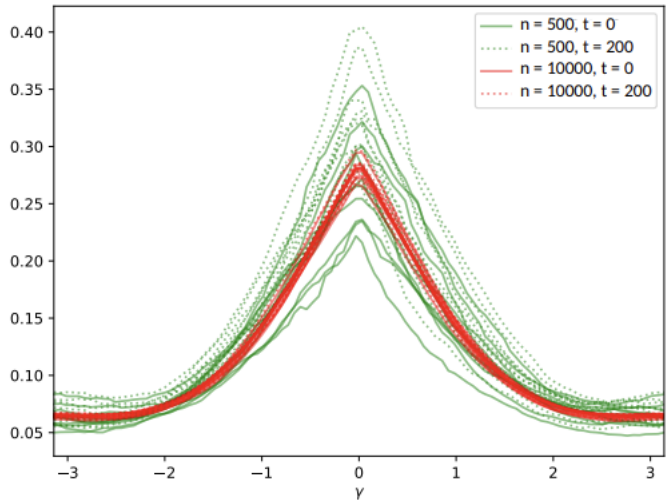


Figure 5: NTK Convergence. Source [JGH20]
Infinite-width stability check for pairs of (neurons size, iterations).

- initialization $t = 0$ and $T = 200$ for 200 steps with learning rate 1.0
- one datapoint as pivot $x_0 = (1, 0)$
- points $x = (\cos(\gamma), \sin(\gamma))$ on the unit circle
- $f^*(x_1, x_2) = x_1 x_2$
- least squares cost as in Subsection IV.2
- random $\mathcal{N}(0, 1)$ parameters

The combinations of information are fed into the NTK $\boldsymbol{\Theta}^{(L)}$ which is plotted in Figure 5. On the horizontal axis the γ angle is associated with higher similarity on the vertical axis at the center, as expected. Infact, the pivot is $x_0 = (1, 0)$ which has a representation for $\gamma = 0$. We also find that more iterations lead to a higher concentration of the similarity distribution around the target, which is a learning phenomenon. It is easily recognized that the red curves have less variance and less learning with respect to the green ones. This inflating phenomenon associated to the smaller network is less common at high dimensions where **lazy training** appears (see Subsection IV.1).

V.2 Regression

We choose:

- $L = 4$ as before
- $f_{\theta(t)}$ for $t \to \infty$ with its expected distribution at the infinite-width limit
- $f_{\theta(T)}$ for $T = 1000 \gg 1$ with its computed distribution
- widths for $f_{\theta(T)}$ $n \in \{50, 1000\}$
- 10 random initializations
- $N = 4$ points on the unit circle at training
- 1000 steps at learning rate 1.0, thus the $T = 1000$

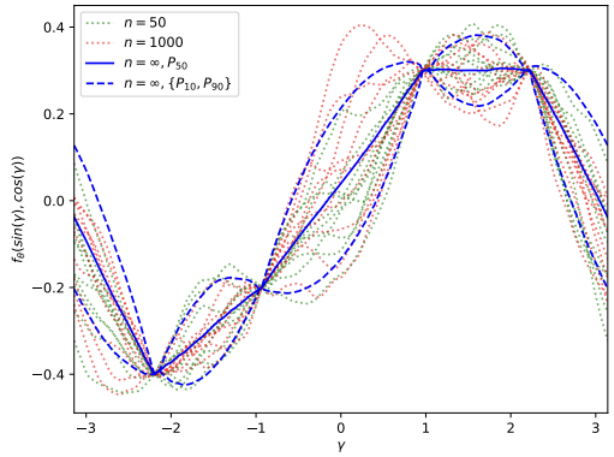


Figure 6: Regression check. Source [JGH20]
The blue lines are an estimation of the asymptotic Gaussian, the red and green lines are real dynamics. Theoretically, red and gree should tend to the blues.

By the discussion of Subsection IV.2, we know that $f_{\theta(t)}$ is Gaussian at any time point of evaluation. The kernels and covariances at the limit $\boldsymbol{\Theta}^{(4)}_{\infty}, \Sigma^{(4)}$ are approximated via a large width network $n = 10000 \gg 100$ and used to calculate three quantiles $\{0.1, 0.5, 0.9\}$ of the limiting Gaussian $\lim_{t\to\infty} f_{\theta(t)}$. The result is shown in Figure 6. It is easily noticed that already at $n = 50$ the approximation of $\lim_{t\to\infty} f_{\theta(t)}$ is good.

V.3 Principal Components

As argued in Subsection IV.2 we could equivalently split the dynamics along principal components of the kernel when performing gradient descent. Following this idea, if we impose that the starting function is just different from the target on one **abstract direction**, we should be able to concentrate the dynamics to it and observe a clear exponential convergence rate.
For this purpose, the authors take the `MNIST` dataset, a classic but easy one among the benchmarks. Images have input dimension $n_0 = 28 \times 28 = 724$, labels have scalar dimension $y \in \{0, \dots, 9\} \subset \mathbb{R}$ so $n_L = 1$.
To compute the principal components of the Π map from Equation IV.2.3 a batch (a subsample) of size $N = 512$ is inspected with a network with $n = 10000$ to approximate the infinite-width limit. The method used to find the eigendecomposition is Power iteration [12]. Three eigenvalues are found: $\lambda_1 = 0.0457, \lambda_2 = 0.00108, \lambda_3 = 0.00078$.

Observation V.3.1 (About the decomposition)**.** *The authors stress that since the Kernel PCA is not centered, the first component is almost equal to a constant function. Intuitively, by not centering, we are also considering the bias inside the model. In this setting, the choice of $\beta = 0.1$ is instrumental since for a bigger β the gap between λ_1 and λ_2 would have been bigger, harming the dynamics of learning [JGH20].*

[12] also known as Von Mises Algorithm.

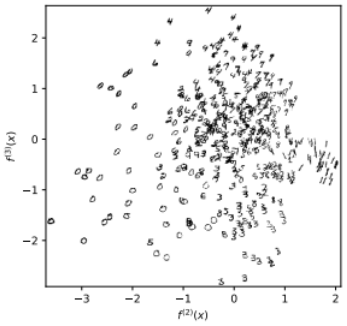


Figure 7: `MNIST` Dataset Visualization [JGH20]
The second and third principal components are plotted to show their variation. The points are *nicely* set on the plane with their number realization.

Observation V.3.2 (On the power iteration method)**.** *Given a matrix A, the power iteration method only finds the eigenpair associated to the largest eigenvalue. Nevertheless, we can simply iterate and find the subsequent ones by subtracting the eigenspace. Assuming $(\lambda, \vec{\boldsymbol{v}})$ is the output, we rerun the procedure on:*

$$A' = A - \lambda \vec{\boldsymbol{v}} \, (\vec{\boldsymbol{v}})^T$$

Given f_0, the first gap is $f^* - f_0$. If this gap is exclusively aligned with one of the eigenfunctions $f^{(i)}$, where by aligned we mean equal or proportional, then the learning dynamics would ideally focus on such a direction with a convergence rate of $e^{-\lambda_i}$ at each $t = 1$ iteration.
This hypothesis is checked against ANNs equipped with a function to estimate of the form:

$$f^* = f_{\theta(0)} + \frac{1}{2} f^{(2)}$$

so that the difference start-target is concentrated on the second eigenspace. For each iteration until convergence, the updated difference is expressed as:

$$f_{\theta(t)} - f^* = g_t + h_t \quad g_t \propto f^{(2)}, \; h_t \perp f^{(2)}$$

In the infinite-width limit, we are sure that:

$$\|g_t\|_{p^{in}} = \frac{1}{2} e^{-\lambda_2 t} \quad \|h_t\|_{p^{in}} = 0$$

with exponential convergence on the parallel direction, and null dynamics on the perpendicular direction.
This behavior is also observed in real networks with $n \in \{100, 1000, 10000\}$, that present a positive trend toward the hypothesis for both norms over time, as Figures 8, 9 suggest.

Observation V.3.3 (On the speed of convergence)**.** *Empirically, with equal learning rates, smaller in n networks are faster at convergence. This pheonomenon has to be commented with caution, as in principle a larger network is more stable and would bear a larger learning rate. In other words, there is competition between the size of the hidden neurons n and the learning rate. Such inflation of convergence was also mentioned in the comment of Figure 5 for the smaller network.*

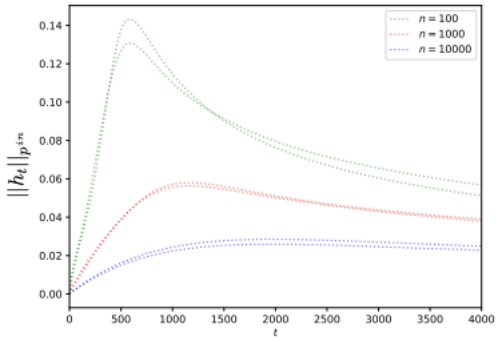


Figure 8: Principal Components. Source [JGH20]
The dynamics of h_θ tend to nullity as the neurons increase in number. Here the null curve is not plotted but is just the 0 line.

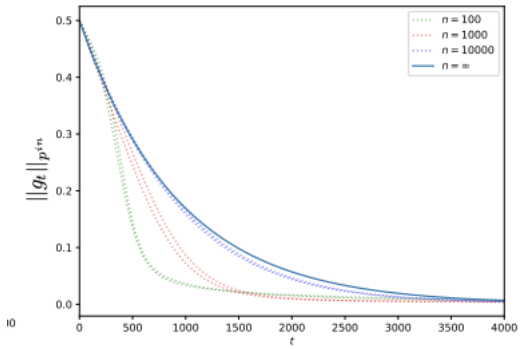


Figure 9: Principal Components. Source [JGH20]
The dynamics of g_θ tend to exponential as the neurons increase in number.

VI Conclusion and Further Directions

We showed that a large class of Artificial Neural Networks at the divergent limit of all hidden layers, follows the dynamics of a kernel: the Neural Tangent Kernel. Such object is a description of local dynamics of gradient descent performed on the architecture at the function-level space. Provided that the NTK dominated dynamics converge subject to positive definiteness of the kernel, it is also the case that this condition implies that the infinite-width limit of an ANN will converge. This last observation also suggests an interesting connection with early stopping heuristics.

The open questions are:

- how well can a Kernel method work?
- is Deep Learning ultimately doing kernels?
- if DL is only doing kernels, is it good or bad?

In practice, there is still a significant gap compared to deep learning, State Of The Art (SOTA) is not reached by kernel methods, even if based on NTKs. NTKs improve performance in general, but do not reach SOTA. Among the theoretical works that attempt to close this gap, we briefly mention [LMZ20; Kar+21]. A nice exploration of code implementations is given in [Aro+22; VBN22]. We avoid discussing their contents and provide just references for the interested reader. Instead, we quickly provide an Example and two interesting visualizations from other works.

If we do not care about performance but only the statistical aspect, NNs do better than kernels. Below we propose an Example.

Example VI.0.1 (NTK vs. NN, [Ten22b])**.** *Here, NTK or any kernel method is statistically limited, while NNs are not. The intuition is that the kernel has fixed features in the NTK approach. Consider* $y \in \pm 1$ *and* $x_i \in \{-1,+1\}^{n_0} = \mathscr{X}$ *iid. We are working with a vector of* ± 1 *binary inputs of length* n_0*. If* $y^* = x_1 \cdot x_2$*, it is not linearly separable, and we need a non-linear model. Use an NN with one layer as we saw in Subsection II.2. Add a regularization of the* L^2*-norm which is equivalent to regularizing*[13]

$$\sum_j |b_j| \, \|\vec{\boldsymbol{a}}_j\|_2$$

The best solution, with minimum norm is a sparse combination of neurons [Ten22b]. In this case, for an input $x = (x_1, x_2, \ldots, x_{n_0})$*, the estimation is exactly computed as:*

$$\widehat{y} = \frac{1}{2}\left[\texttt{ReLU}(x_1 + x_2) + \texttt{ReLU}(-x_1 - x_2) - \texttt{Relu}(x_1 - x_2) - \texttt{Relu}(x_2 - x_1)\right]$$

this is equal to the function since $\texttt{ReLu}(t) + \texttt{ReLu}(-t) = |t|$ *and :*

$$\widehat{y} = \frac{1}{2}(|x_1 + x_2| - |x_1 - x_2|) = x_1 x_2$$

It is sufficient to implement just 4 `ReLu` *neurons to solve this task.*
If we chose to use an NTK, we would not learn any features and try to make a dense combination of existing features. Infact we initialize the weights at random locations and thus make a randomly weighted combination, which is not sparse.

$$\widehat{y_i} = \theta^T \varphi(x_i)$$

[13] also known as the path norm

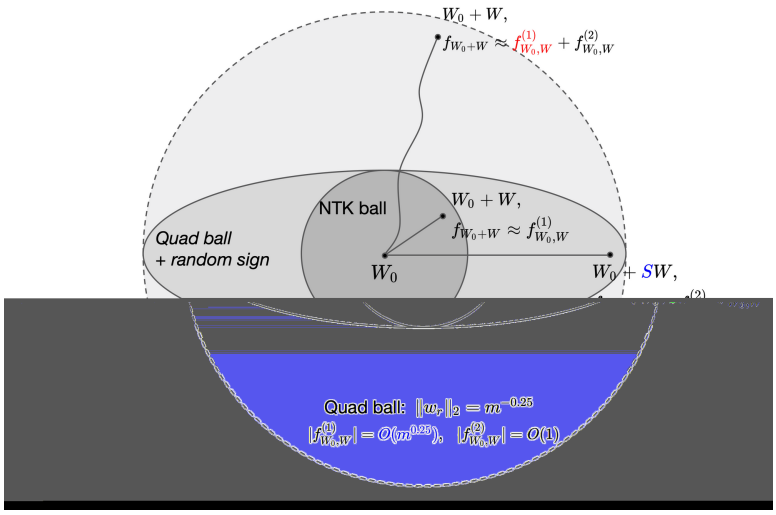


Figure 10: Beyond NTK. Source [Bai21]
The **notation is different**

where the feature map uses all the dimensions, instead of the NN using just the first specialized two.

Theorem VI.0.2 (Informal conclusion, [Ten22b])**.** *Kernel methods with NTK requires* $n = \Omega(d^2)$ *samples to learn the problem. In contrast, a regularized NN only needs* $n = O(d)$ *samples.*

VI.1 Interesting ideas

We present two potential further topics with a graphically impactful result.

Inspecting the higher order terms of a Taylor expansion is nice and intuitive. A work in this direction is [BL20], with also a more accessible blog post [Bai21]. Ultimately, a larger class of descriptive methods for Neural Networks is found. For an idea, without matching notation, see Figure 10.

With quadratic scaling of parameters with respect to dataset size it was also observed that in the over-parametrized regime NTK descriptions of NNs experience a triple descent (or even more). While double descent is commonly associated with over-parametrization, this proposal has the objective to understand further the phenomenon with a precise quantification of the pace changes in the test error. For an idea of the result, see Figure 11.

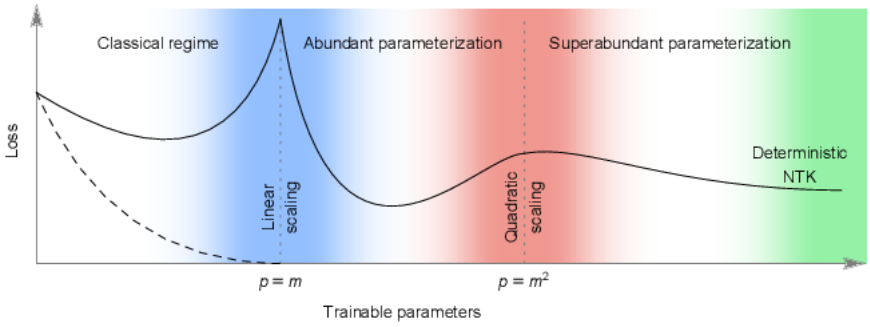


Figure 11: Triple Descent. Source [AP20]
The **notation is different**

References

[JGH20] Arthur Jacot, Franck Gabriel, and Clément Hongler. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. Feb. 2020. arXiv: 1806.07572 [cs, math, stat].

[CPW21] Lei Chu, Hao Pan, and Wenping Wang. *Unsupervised Shape Completion via Deep Prior in the Neural Tangent Kernel Perspective*. Apr. 2021. DOI: 10.48550/arXiv.2104.09023. arXiv: 2104.09023 [cs].

[Soh20] Soheil Feizi. *Lecture 7 - Deep Learning Foundations: Neural Tangent Kernels*. Sept. 2020.

[Ten22a] Tengyu Ma. *Stanford CS229M - Lecture 13: Neural Tangent Kernel*. Nov. 2022.

[Ten22b] Tengyu Ma. *Stanford CS229M - Lecture 14: Neural Tangent Kernel, Implicit Regularization of Gradient Descent*. Nov. 2022.

[Vad19] Rajat Vadiraj Dwaraknath. *Understanding the Neural Tangent Kernel*. 2019.

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

[Wal21] Neil Walton. *Neural Tangent Kernel*. Mar. 2021.

[Wen22] Lilian Weng. *Some Math behind Neural Tangent Kernel*. https://lilianweng.github.io/posts/2022-09-08-ntk/. Sept. 2022.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8.

[Les+93] Moshe Leshno et al. "Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function". In: *Neural Networks* 6.6 (Jan. 1993), pp. 861–867. ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80131-5.

[Dau+14] Yann Dauphin et al. *Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization*. June 2014. DOI: 10.48550/arXiv.1406.2572. arXiv: 1406.2572 [cs, math, stat].

[Pas+14] Razvan Pascanu et al. *On the Saddle Point Problem for Non-Convex Optimization*. May 2014. DOI: 10.48550/arXiv.1405.4604. arXiv: 1405.4604 [cs].

[Cho+15] Anna Choromanska et al. *The Loss Surfaces of Multilayer Networks*. Jan. 2015. DOI: 10.48550/arXiv.1412.0233. arXiv: 1412.0233 [cs].

[PB17] Jeffrey Pennington and Yasaman Bahri. "Geometry of Neural Network Loss Surfaces via Random Matrix Theory". In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, July 2017, pp. 2798–2806.

[KAA19] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. *Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach*. Oct. 2019. DOI: 10.48550/arXiv.1806.01316. arXiv: 1806.01316 [cond-mat, stat].

[MMN18] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. "A Mean Field View of the Landscape of Two-Layer Neural Networks". In: *Proceedings of the National Academy of Sciences* 115.33 (Aug. 2018), E7665–E7671. DOI: 10.1073/pnas.1806579115.

[Sag+18] Levent Sagun et al. *Empirical Analysis of the Hessian of Over-Parametrized Neural Networks*. May 2018. DOI: 10.48550/arXiv.1706.04454. arXiv: 1706.04454 [cs].

[Zha+17] Chiyuan Zhang et al. *Understanding Deep Learning Requires Rethinking Generalization*. Feb. 2017. DOI: 10.48550/arXiv.1611.03530. arXiv: 1611.03530 [cs].

[BMM18] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. *To Understand Deep Learning We Need to Understand Kernel Learning*. June 2018. DOI: 10.48550/arXiv.1802.01396. arXiv: 1802.01396 [cs, stat].

[Nea96] Radford M. Neal. *Bayesian Learning for Neural Networks*. Ed. by P. Bickel et al. Vol. 118. Lecture Notes in Statistics. New York, NY: Springer New York, 1996. ISBN: 978-0-387-94724-2 978-1-4612-0745-0. DOI: 10.1007/978-1-4612-0745-0.

[DFS17] Amit Daniely, Roy Frostig, and Yoram Singer. *Toward Deeper Understanding of Neural Networks: The Power of Initialization and a Dual View on Expressivity*. May 2017. DOI: 10.48550/arXiv.1602.05897. arXiv: 1602.05897 [cs, stat].

[Mat17] A. G. D. G. Matthews. "Sample-Then-Optimize Posterior Sampling for Bayesian Linear Models". In: 2017.

[Lee+18] Jaehoon Lee et al. *Deep Neural Networks as Gaussian Processes*. Mar. 2018. DOI: 10.48550/arXiv.1711.00165. arXiv: 1711.00165 [cs, stat].

[Mat+18] Alexander G. de G. Matthews et al. *Gaussian Process Behaviour in Wide Deep Neural Networks*. Aug. 2018. DOI: 10.48550/arXiv.1804.11271. arXiv: 1804.11271 [cs, stat].

[CS09] Youngmin Cho and Lawrence Saul. "Kernel Methods for Deep Learning". In: *Advances in Neural Information Processing Systems*. Vol. 22. Curran Associates, Inc., 2009.

[CB18] Lenaic Chizat and Francis Bach. *On the Global Convergence of Gradient Descent for Over-parameterized Models Using Optimal Transport*. Oct. 2018. DOI: 10.48550/arXiv.1805.09545. arXiv: 1805.09545 [cs, math, stat].

[Bro+21] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. May 2021. DOI: 10.48550/arXiv.2104.13478. arXiv: 2104.13478 [cs, stat].

[LeC+12] Yann A. LeCun et al. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Vol. 7700. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35288-1 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3.

[SC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. First. Cambridge University Press, June 2004. ISBN: 978-0-521-81397-6 978-0-511-80968-2. DOI: 10.1017/CBO9780511809682.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer New York, 2009. ISBN: 978-0-387-84857-0 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7.

[Gho+21] Benyamin Ghojogh et al. *Reproducing Kernel Hilbert Space, Mercer's Theorem, Eigenfunctions, Nystr\"om Method, and Use of Kernels in Machine Learning: Tutorial and Survey*. June 2021. arXiv: 2106.08443 [cs, math, stat].

[Cov65] Thomas M. Cover. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3 (June 1965), pp. 326–334. ISSN: 0367-7508. DOI: 10.1109/PGEC.1965.264137.

[MMR97] Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks*. Complex Adaptive Systems. Cambridge, Mass: MIT Press, 1997. ISBN: 978-0-262-13328-9.

[Den+09] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[SSM97] Bernhard Schölkopf, Alexander Smola, and Klaus Robert Müller. "Kernel Principal Component Analysis: 7th International Conference on Artificial Neural Networks, ICANN 1997". In: *Artificial Neural Networks - ICANN 1997 - 7th International Conference, Proceeedings*. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (1997). Ed. by Wulfram Gerstner et al., pp. 583–588. ISSN: 3540636315. DOI: 10.1007/bfb0020217.

[Bac20] Francis Bach. *Effortless Optimization through Gradient Flows – Machine Learning Research Blog*. 2020.

[Aro+19] Sanjeev Arora et al. *On Exact Computation with an Infinitely Wide Neural Net*. Nov. 2019. arXiv: 1904.11955 [cs, stat].

[Dai+22] Zhongxiang Dai et al. *Sample-Then-Optimize Batch Neural Thompson Sampling*. Oct. 2022. DOI: 10.48550/arXiv.2210.06850. arXiv: 2210.06850 [cs].

[Goo+14] Ian J. Goodfellow et al. *Generative Adversarial Networks*. June 2014. DOI: 10.48550/arXiv.1406.2661. arXiv: 1406.2661 [cs, stat].

[Old+09] Keith B. Oldham et al. *An Atlas of Functions: With Equator, the Atlas Function Calculator*. Mathematics and Statistics. New York, NY: Springer US Springer e-books, 2009. ISBN: 978-0-387-48807-3.

[Gne13] Tilmann Gneiting. "Strictly and Non-Strictly Positive Definite Functions on Spheres". In: *Bernoulli* 19.4 (Sept. 2013). ISSN: 1350-7265. DOI: 10.3150/12-BEJSP06. arXiv: 1111.7077 [math, stat].

[SSM98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". In: *Neural Computation* 10.5 (July 1998), pp. 1299–1319. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976698300017467.

[Cha+12] Venkat Chandrasekaran et al. "The Convex Geometry of Linear Inverse Problems". In: *Foundations of Computational Mathematics* 12.6 (Dec. 2012), pp. 805–849. ISSN: 1615-3375, 1615-3383. DOI: 10.1007/s10208-012-9135-7. arXiv: 1012.0621 [math, stat].

[LMZ20] Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. *Learning Over-Parametrized Two-Layer ReLU Neural Networks beyond NTK*. July 2020. DOI: 10.48550/arXiv.2007.04596. arXiv: 2007.04596 [cs, math, stat].

[Kar+21] Stefani Karp et al. "Local Signal Adaptivity: Provable Feature Learning in Neural Networks Beyond Kernels". In: *Advances in Neural Information Processing Systems*. Oct. 2021.

[Aro+22] Sanjeev Arora et al. "Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks". In: *International Conference on Learning Representations*. Feb. 2022.

[VBN22] Nikhil Vyas, Yamini Bansal, and Preetum Nakkiran. *Limitations of the NTK for Understanding Generalization in Deep Learning*. June 2022. DOI: 10.48550/arXiv.2206.10012. arXiv: 2206.10012 [cs].

[BL20] Yu Bai and Jason D. Lee. *Beyond Linearization: On Quadratic and Higher-Order Approximation of Wide Neural Networks*. Feb. 2020. DOI: 10.48550/arXiv.1910.01619. arXiv: 1910.01619 [cs, math, stat].

[Bai21] Yu Bai. *When Are Neural Networks More Powerful than Neural Tangent Kernels?* Mar. 2021.

[AP20] Ben Adlam and Jeffrey Pennington. *The Neural Tangent Kernel in High Dimensions: Triple Descent and a Multi-Scale Theory of Generalization*. Aug. 2020. DOI: 10.48550/arXiv.2008.06786. arXiv: 2008.06786 [cs, stat].

[Rud91] Walter Rudin. *Functional Analysis*. 2nd ed. International Series in Pure and Applied Mathematics. New York: McGraw-Hill, 1991. ISBN: 978-0-07-054236-5.

[CG21] Piermarco Cannarsa and Filippo Gazzola. *Dynamic Optimization for Beginners: With Prerequisites and Applications*. First. EMS Press, Oct. 2021. ISBN: 978-3-9854701-2-9 978-3-9854751-2-4. DOI: 10.4171/etb/23.

[Hun80] Thomas W. Hungerford. *Algebra*. Vol. 73. Graduate Texts in Mathematics. New York, NY: Springer, 1980. ISBN: 978-1-4612-6103-2 978-1-4612-6101-8. DOI: 10.1007/978-1-4612-6101-8.

[GKD19] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. "A Visual Exploration of Gaussian Processes". In: *Distill* 4.4 (Apr. 2019), e17. ISSN: 2476-0757. DOI: 10.23915/distill.00017.

[AFP00] Luigi Ambrosio, Nicola Fusco, and Diego Pallara. *Functions of Bounded Variation and Free Discontinuity Problems*. Oxford Mathematical Monographs. Oxford ; New York: Clarendon Press, 2000. ISBN: 978-0-19-850245-6.

A Required Notions

In the Appendix we present the basic notions and Lemmas that are needed but would have harmed the flow of the exposition. Additional references are also mentioned in case the reader needs them.

A.1 Algebra

Definition A.1.1 (Fields and vector spaces)**.** *We use the classical definition of Field, as a triplet $\mathscr{F} = (F, \oplus, \odot)$ formed by a set and well-defined sum and multiplication operations on it. Accordingly, a vector space $\mathscr{V} = (V, \boxplus, \boxtimes)$ on a field $\mathscr{F}$ is a triplet with well defined operations in the classical sense.*
For simplicity, most of the times the symbols F, V are used to refer to $\mathscr{F}, \mathscr{V}$.

Example A.1.2 (The vector space we need)**.** *$\mathcal{F} = \{f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}\}$ is a vector space when paired with the intuitive operations on it.*

Definition A.1.3 (Linear operator)**.** *For two vector spaces W, V a linear operator is a map:*

$$L : V \to W$$

such that the additivity and homogeneity are preserved, namely:

- $L(\vec{\boldsymbol{v_1}} + \vec{\boldsymbol{v_2}}) = L(\vec{\boldsymbol{v_1}}) + L(\vec{\boldsymbol{v_2}})$
- $L(c\vec{\boldsymbol{v_1}}) = cL(\vec{\boldsymbol{v_1}})$

Definition A.1.4 (Bilinear map)**.** *For three vector spaces V, W, X a bilinear map is a function:*

$$B : V \times W \to X$$

such that:

- *$v \to B(v, w)$ is linear from V to X for all $w \in W$*
- *$w \to B(v, w)$ is linear from W to X for all $v \in V$*

Definition A.1.5 (Bilinear form)**.** *A bilinear form is a bilinear map of a vector space mapping to the field over which it is defined:*

$$B : V \times V \to F$$

Example A.1.6 (Inner product)**.** *In our context, it will be the case that $\mathscr{F} = (\mathbb{R}, +, \cdot)$, and $\mathscr{V} = (\mathbb{R}^n, +, \cdot)$ where the operations are the classical ones and do not need explanations. Then, the most intuitive example for a bilinear form is the inner product:*

$$\langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{w}} \rangle = \vec{\boldsymbol{v}}^T \vec{\boldsymbol{w}} = \sum_{i=1}^{n} v_i w_i \in \mathbb{R}$$

Throughout the discussion, we will use different notions of inner products, but the idea is always the same.

Definition A.1.7 (Positive definiteness, and other conditions)**.** *A bilinear form B over a field $\mathbb{R}$ is:*

- *positive semi-definite (p.s.d.) if* $B(\vec{\boldsymbol{v}}, \vec{\boldsymbol{v}}) \geq 0$
- *positive definite (p.d.) if* $B(\vec{\boldsymbol{v}}, \vec{\boldsymbol{v}}) > 0$
- *symmetric if* $B(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) = B(\vec{\boldsymbol{w}}, \vec{\boldsymbol{v}}$

Other easy notions are included in the usual textbooks. A classic reference is [Rud91]. A shorter one that might be sufficient is [CG21](Sec. 2).

Definition A.1.8 (Operator norm)**.** *For two normed vector spaces* $(V, \|\cdot\|_V)$ *and* $(W, \|\cdot\|_W)$ *a linear operator* $L : V \to W$ *has norm:*

$$\|L\|_{op} = \sup \{\|L\vec{\boldsymbol{v}}\|_W : \|\vec{\boldsymbol{v}}\|_V \leq 1\}$$

Lemma A.1.9 (Operator norm bound)**.** *Assume* L *is a linear operator inner product spaces* $(V, \langle\cdot,\cdot\rangle_V)$ *and* $(W, \langle\cdot,\cdot\rangle_W)$*. If the column norms of* L *are bounded, then the operator norm is bounded.*

Proof. It is sufficient to recognize that an inner product defines naturally a norm[14] via $\|\cdot\| = \sqrt{\langle\cdot,\cdot\rangle}$. With the notion of inner product, we can express matrix multiplication in terms of sum of inner product of the rows and conclude the claim. Mathematically:

$$\begin{aligned}
\|L\vec{\boldsymbol{v}}\|_W &= \left\|\sum \langle L_i, \vec{\boldsymbol{v}}\rangle_V\right\| \\
&\leq \sum \|\langle L_i, \vec{\boldsymbol{v}}\rangle\| && \text{triangle inequality} \\
&\leq \sum \|L_i\| \, \|u\| && \text{Cauchy-Schwarz} \\
&\leq \sum \|L_i\| && \|u\| \leq 1 \\
&< C \quad C \in \mathbb{R} && \text{hypothesis}
\end{aligned}$$

and the operator norm is bounded. □

Definition A.1.10 (Dual space)**.** *Given a vector space* V *over a field* F *the dual, denoted as* V^* *is the set of linear forms on* V*, namely functions:*

$$D : V \to F$$

Example A.1.11 (Half fixed bilinear form is a dual element)**.** *Let* V, W *be vector spaces over a field* F*, and* B *a bilinear form. Consider* $v \in V$ *to be fixed. Then, the function:*

$$D : W \to F \quad w \to D(w) = B(v, w)$$

is an element of the dual space of W*, namely* $D \in W^*$*.*

Definition A.1.12 (Semi-norm)**.** *Differently from a norm, a seminorm over a real-valued vector space is a function:*

$$p : V \to \mathbb{R}$$

such that:

- *(triangle inequality)* $p(\vec{\boldsymbol{v}} + \vec{\boldsymbol{w}}) \leq p(\vec{\boldsymbol{v}}) + p(\vec{\boldsymbol{w}})$
- *(absolute homogeneity)* $p(s\vec{\boldsymbol{v}}) = |s|p(\vec{\boldsymbol{v}})$

[14] the other way is not true, so we assume we have an inner product just to be sure

The idea is that the two conditions imply $p(\boldsymbol{0}) = 0$ but do not satisfy the separation property that $p(\vec{\boldsymbol{v}}) = 0 \iff \vec{\boldsymbol{v}} = \boldsymbol{0}$, which is the third requirement in the Definition of a norm.

Definition A.1.13 (Tensor and tensor product)**.** *Consider two vector spaces V, W over the same field. Their tensor product $V \otimes W$ is a vector space obtained via a bilinear map:*

$$B : V \times W \to V \otimes W$$

We call tensor product of two vectors an element $\vec{\boldsymbol{v}} \otimes \vec{\boldsymbol{w}} \in V \otimes W$ identified by such map. The constructions are many, and are equal up to isomorphisms.
The Tensor product can be seen as a generalized outer product.

Example A.1.14 (Tensor product space construction)**.** *Consider the setting presented in Section I. This allows us to construct the $\mathcal{F} \otimes \mathcal{F}$ tensor product as follows. Denote the basis of $\mathcal{F}$ as $\mathscr{B}$. The space $\mathcal{F} \otimes \mathcal{F}$ has basis elements $f, g \in \mathscr{B}$. $\mathcal{F} \otimes \mathcal{F}$ is a set of functions of the form:*

$$T : \mathscr{B} \times \mathscr{B} \to \mathbb{R}$$

which have a finite number of non-zero values. Namely, pairs $f, g \in \mathcal{F} \times \mathcal{F}$ such that $T(f, g) \neq 0$ are finite. Classic operations make $\mathcal{F} \otimes \mathcal{F}$ a vector space.
The function such that two specific basis elements $(f, g) \in \mathscr{B} \times \mathscr{B}$ map to 1 and all other combinations map to zero is denoted as $f \otimes g$.
Then, the set:

$$\{f \otimes g \mid f, g \in \mathscr{B}\}$$

is a basis for $\mathcal{F} \otimes \mathcal{F}$. The tensor product of two functions is then unambiguously expressed depending on the choice of the basis as:

$$f \otimes g = \underbrace{\left(\sum_{b \in \mathscr{B}} f_b \cdot b\right)}_{=f} \otimes \underbrace{\left(\sum_{b' \in \mathscr{B}} g_{b'} \cdot b'\right)}_{=g} = \sum_{b \in \mathscr{B}} \sum_{b' \in \mathscr{B}} f_b g_{b'} b \otimes b'$$

The Machine Learning common introductory interpretation of a tensor is that of a generalized matrix to multiple dimensions (e.g. a 3D cube), but is not accurate. We give here a constructive definition of tensor product and thus of tensor as a multilinear map. A good reference for a more rigorous treatment is [Hun80].

A.2 Functional Analysis

Definition A.2.1 (Lipschitz function)**.** *a function $\sigma : \mathbb{R} \to \mathbb{R}$ is Lipschitz whenever:*

$$\exists c \in \mathbb{R} \ : \ |f(x) - f(x')| \leq c|x - x'| \quad \forall x, x' \in \mathbb{R}$$

Lemma A.2.2 (Lipschitz functions have bounded derivative)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be c-Lipschitz. Then:*

$$\dot{f}(x) \leq c \quad \forall x \in \mathbb{R}$$

Proof. The result is definitional. We have:

$$|f(x') - f(x)| \leq c|x' - x| \overset{x'=x+h}{\Longrightarrow} \left|\frac{f(x+h) - f(x)}{h}\right| \leq c \overset{h\to 0}{\Longrightarrow} \dot{f}(x) \leq c$$

□

Definition A.2.3 (Functional derivative)**.** *We give the classical definition of functional derivative, restricted to a sufficient formalization for the purpose of this document.*
Consider a functional $C : \mathcal{F} \to \mathbb{R}$, belonging to the dual of some space $\mathcal{F}$ equipped with an inner product $\langle \cdot, \cdot \rangle$. Then, the functional derivative is defined via:

$$\left\langle \frac{\delta C}{\delta \rho}, \phi \right\rangle = \lim_{\epsilon \downarrow 0} \frac{C(\rho + \epsilon\phi) - C(\rho)}{\epsilon} = \left[\frac{d}{d\epsilon} C(\rho + \epsilon\phi)\right]_{\epsilon=0} \qquad \phi \in \mathcal{F}$$

Without going much deeper into the topic, we stress that this generalization is well behaved and inherits most of the properties of classical derivatives. A very nice one that we will use throughout is the chain rule. In particular, for two functionals C, Q

$$\frac{\delta C[Q(\rho)]}{\delta \rho} = \left\langle \frac{\delta C[Q]}{\delta Q}|_{Q=Q(\rho)}, \frac{\delta Q[\rho]}{\delta \rho} \right\rangle$$

For more context, useful examples, and a wider view, a suggestion is [CG21](Sec. 4)

Definition A.2.4 (Positive definite Kernel)**.** *A symmetric function:*

$$K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$$

is a positive definite kernel, not with respect to a norm when for any choice $\vec{\boldsymbol{c}} \in \mathbb{R}^n, \{\vec{\boldsymbol{x}}_i\}_{i=1}^n, \forall n$ it holds that:

$$\sum_i \sum_j c_i c_j K(x_i, x_j) \geq 0$$

Example A.2.5 (Positive definite matrix as a special case of positive definite kernel)**.** *Consider a positive definite Kernel K, then the matrix arising from the p.d. kernel evaluated on the elements of a N-dimensional dataset $\mathcal{D}$ with $x \in \mathbb{R}^d$. Then the matrix:*

$$\widetilde{K}_{ij} = K(x_i, x_j) \in \mathbb{R}^{N \times d}$$

is positive definite in the sense of matrices by construction.

Definition A.2.6 (Gaussian Process)**.** *A stochastic process $(X_t)_{t\in\mathbb{T}}$ is a Gaussian Process if and only if for any choice of $\{t_i\}_{i=1}^n \subset \mathbb{T}$ we have:*

$$(X_{t_1}, \ldots, X_{t_n}) \sim \mathcal{N}^n(\mu_{\vec{\boldsymbol{t}}}, \Sigma_{\vec{\boldsymbol{t}}})$$

This defining property, via a discussion on characteristic functions, allows us to state that Gaussian Processes are completely identified by two functions:

- *a mean $m : \mathbb{T} \to \mathbb{R}$*
- *a covariance function $\Sigma : \mathbb{T} \times \mathbb{T} \to \mathbb{R}$, a positive definite Kernel in the sense of Definition A.2.4*

We then say that $(X_t)_{t\in\mathbb{T}} \sim \mathcal{GP}(m, \Sigma)$.
More intuition about Gaussian processes is on this famous blog page [GKD19].

Theorem A.2.7 (Rademacher Theorem)**.** *For* $U \subset \mathbb{R}^n$ *open and* $f : U \to \mathbb{R}^m$ *Lipschitz* f *is almost everywhere differentiable.*

Proof. [AFP00](Thm. 2.14) or this link for further references and a short proof. □

Definition A.2.8 (Uniform convergence $\rightrightarrows$)**.** *A sequence* $(f_n)_{n\in\mathbb{N}}$ *such that* $f_n : E \to \mathbb{R}$ *is uniformly convergent to* $f : E \to \mathbb{R}$ *if:*

$$\forall \epsilon > 0 \exists N \in \mathbb{N} \text{ such that } \forall n \geq N, \forall x \in E \quad |f_n(x) - f(x)| < \epsilon$$

Namely, there is a common bound on the distance after sufficiently large n*. It is often written as* $f_n \rightrightarrows f$.

Definition A.2.9 (Dense subset)**.** *For a metric space* (X, m) *where the metric is induced by a norm, a subspace* S *is dense if its closure it the whole space:*

$$\overline{S} = X$$

without going too much into the details, we say that for any neighborhood of a point in X *the intersection with* S *will be non empty. Loosely, this means that* A *approximates the elements of* X *with arbitrary precision. In other works, for every element of* X *there exists a sequence* $(s_n) \subset S$ *such that:*

$$\lim_{n\to\infty} \|s_n - x\| = 0$$

Definition A.2.10 (Exponential of a map)**.** *For a map* $\Pi : \mathcal{F} \to \mathcal{F}$ *we define its exponential in a Taylor represnetation sense:*

$$e^{t\Pi}(f) : \mathcal{F} \to \mathcal{F} \quad f \to \sum_{k=0}^{\infty} \frac{t^k}{k!} \Pi^k(f)$$

Definition A.2.11 (Eigenfunctions)**.** *for a linear operator* L *between vector spaces* V, W *the eigenfunctions are a special case of eigenvectors, with associated eigenvalues. Namely,* (λ, f) *is an eigenvalue pair when:*

$$Lf = \lambda f, \quad \lambda \neq 0$$

Definition A.2.12 (Gram Matrix $\widetilde{\boldsymbol{K}}$)**.** *A Gram matrix in the classical sense is obtained from a set of vectors* $\{\vec{\boldsymbol{v_i}}\}_{i=1}^N$ *in an inner product space* $(V, \langle\cdot,\cdot\rangle$ *as:*

$$\widetilde{\boldsymbol{K}} \in \mathbb{R}^{N\times N} \quad \widetilde{\boldsymbol{K}}_{\boldsymbol{ij}} = \langle \vec{\boldsymbol{v_i}}, \vec{\boldsymbol{v_j}} \rangle$$

It can be shown that such matrix is positive semi-definite if and only if the kernel that defines it is valid, in the sense that it identifies an inner product in a higher dimensional space via a transformation $\phi : \mathscr{X} \to \mathbb{R}^D$ *where* $|\mathscr{X}| = d \ll D$*. It is also equivalent to requiring that the kernel is symmetric positive semi-definite. This well-known result is a special case of Mercer's Theorem. The statement and a proof can be found in literature [SC04](Section 3).*
In the context of this document, we work with multidimensional kernels, but this notion naturally extends with a multidimensional kernel $\boldsymbol{K}$ *identifying a Gram matrix of the form:*

$$\widetilde{\boldsymbol{K}} \in \mathbb{R}^{Nn_L\times Nn_L} \quad \left(\widetilde{\boldsymbol{K}}\right) = (\boldsymbol{K_{kk'}}(x_i, x_j))_{ik,jk'} \quad i, j \in 1, \ldots, n_0 \quad k, k' \in 1, \ldots n_L$$

Proposition A.2.13 (Gram Kernel connection)**.** *A kernel $\boldsymbol{K}$ is positive definite in the sense of Definition I.3.6 if and only if the Gram matrix $\widetilde{\boldsymbol{K}}$ is invertible (by positive definiteness).*

Lemma A.2.14 (Grownwall's Lemma)**.** *Consider f a real valued continuous function on an interval I, differentiable on the interior I^0 of I. We say the interior is (a, b). Let β be continuous and real valued on I as well. Then:*

$$f'(t) \leq \beta(t) f(t) \implies f(t) \leq f(a) \exp\left\{ \int_a^t \beta(s) ds \right\} \quad \forall t \in I$$

Proof. [CG21](Exercise 8.8), but also the Wikipedia page has a satisfactory proof. □

Lemma A.2.15 (Derivative of norm vs norm of derivative)**.** *We provide an almost general result. The case for null derivatives is advanced.*
For an inner product vector space $(V, \langle \cdot, \cdot \rangle_V)$ and a differentiable function $f : \mathbb{R} \to V$ with non-zero derivative, the derivative of the norm is less than the norm of the derivative:

$$\partial_t \|f\|_V \leq \|\partial_t f\|$$

Proof. By the chain rule:

$$\partial_t \|f\|_V^2 = 2 \|f\|_V \partial_t \|f\|_V$$

However, it is also the case that $\|f\|_V^2 = \langle f, f \rangle_V$. Then:

$$\begin{aligned}
\partial_t \|f\|_V^2 &= \partial_t \langle f, f \rangle_V \\
&= \langle \partial_t f, f \rangle_V + \langle f, \partial_f \rangle_V && \text{derivative of } \langle \cdot, \cdot \rangle \\
&= 2 \langle \partial_t f, f \rangle_V && \text{symmetry of } \langle \cdot, \cdot \rangle_V \\
&\leq 2 \|\partial_t f\|_V \|f\|_V && \text{Cauchy-Schwarz}
\end{aligned}$$

Combining the two and simplyfying $2 \|f\|_V$ we get the claim $\partial_t \|f\|_V \leq \|\partial_t f\|_V$. □