Notes on the Neural Tangent Kernel

A beginners' guide

Simone Maria Giancola[1]

[1]Bocconi University, Milan, Italy

Machine Learning II, Bocconi University, February 2023

Lecture Contents

Lecture Path

Content

- Mostly an exploration of the results of [JGH20]
- additional helpful resources:
  - 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

Content

- Ideally, a sufficient explanation for a beginner
- The doc at this link has the proofs, a wide Appendix section and lots of references (70 pages)

Boxes I

This is a definition

Here I define something

This is a theorem

Something is gnihtemoS backwards

This is an assumption

assumptions are purple boxes

A remark an observation or an example

for example, I observe or remark that this is an observation

Partial Notation

- $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ dataset
- Neural Network layers $\ell \in \{0, \dots, L\}$
- $x_i \in \mathscr{X} \subseteq \mathbb{R}^{n_0}, y_i \in \mathscr{Y} \subseteq \mathbb{R}^{n_L}$
- ∂_t derivative with respect to t
- $\langle \cdot, \cdot \rangle_{p^{in}}, \|\cdot\|_{p^{in}}$ inner product and norm wrt empirical distribution $p^{in} = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}$
- $\theta = \left\{ W^{(\ell)}, b^{(\ell)} \right\}_{\ell=0}^{L-1}$ parameters, $\theta \in \mathbb{R}^P$
- $\mathcal{F} = \{f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}\}$ space of realization functions $f_\theta(x)$
- σ non-linearity
- $\widetilde{\alpha}^{(\ell)}(x;\theta), \alpha^{(\ell)}(x;\theta) = \sigma\left(\widetilde{\alpha}^{(\ell)}(x;\theta)\right)$ preactivation and activation at layer ℓ
- $\mathscr{L}$ dataset loss, $\mathcal{L}$ element-wise loss

Symbols and colors instead of proofs

Some parts are advanced, and time is short. For the sake of the presentation, technnical aspects are left aside, instead we use:

- ☺ means good for what we want to do
- ☹ means bad for what we want to do

Symbols and colors instead of proofs

Some parts are advanced, and time is short. For the sake of the presentation, technnical aspects are left aside, instead we use:

- means difficult, overlooked, taken as granted

The Artificial Neural Network model

We aim to estimate a function of the form:

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

Arising from a fully connected ANN.
The objective is to efficiently approximate $\vec{\mathbf{y}}$ according to a parametric loss $\mathscr{L} : \mathbb{R}^P \to \mathbb{R}_+$.

The Artificial Neural Network model

We aim to estimate a function of the form:

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

Arising from a fully connected ANN.
The objective is to efficiently approximate $\vec{\mathbf{y}}$ according to a parametric loss $\mathscr{L} : \mathbb{R}^P \to \mathbb{R}_+$.

Optimization problem

Solve

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

ANN Graphically

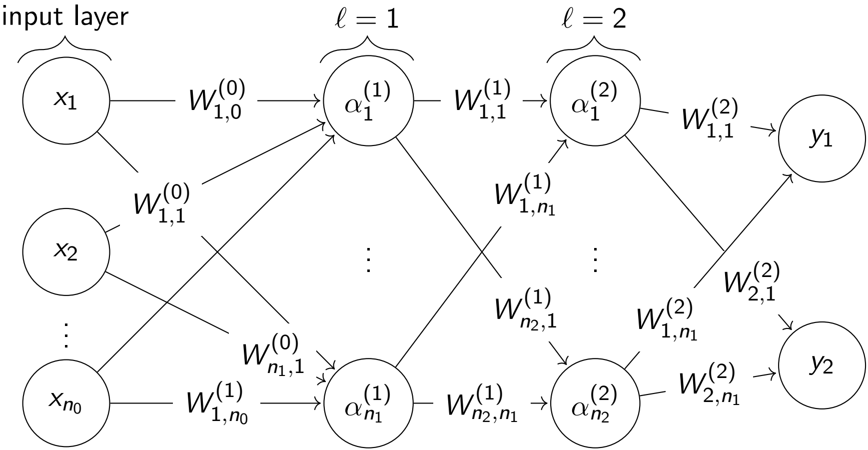

Figure: $L = 3$, **bias omitted**, $\alpha^{(\ell)}$ activations

Neural Network Model, functional view

Realization Function

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

the **network function** is $f_\theta(x) \in \mathcal{F}$.

Functional Cost

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

which can be regression or cross entropy.

Neural Network Model, functional view

Realization Function

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

the **network function** is $f_\theta(x) \in \mathcal{F}$.

Functional Cost

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

which can be regression or cross entropy.

Updated Optimization problem

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

same as old one but at a function level.

ANN, functional rescaled view

For activations and preactivations which for $\ell \in 0, \dots, L$ are of the form:

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

state the recursion:

$$\begin{aligned} \alpha^{(0)}(x;\theta) &= x, \quad \theta_p \sim \mathcal{N}(0,1) \quad \forall p \\ \widetilde{\alpha}^{(\ell+1)}(x;\theta) &= \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \alpha^{(\ell)}(x;\theta) + \beta b^{(\ell)} \quad \beta > 0 \\ \alpha^{(\ell)}(x;\theta) &= \sigma\left(\alpha^{(\ell)}(x;\theta)\right) \end{aligned}$$

We set $f_\theta(x) = \widetilde{\alpha}^{(L)}(x;\theta)$, notice that we specifically use the preactivation to have a final linear combination.

ANN, functional rescaled view

For activations and preactivations which for $\ell \in 0, \ldots, L$ are of the form:

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

state the recursion:

$$\begin{aligned}
\alpha^{(0)}(x;\theta) &= x, \quad \theta_p \sim \mathcal{N}(0,1) \quad \forall p \\
\widetilde{\alpha}^{(\ell+1)}(x;\theta) &= \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \alpha^{(\ell)}(x;\theta) + \beta b^{(\ell)} \quad \beta > 0 \\
\alpha^{(\ell)}(x;\theta) &= \sigma\left(\alpha^{(\ell)}(x;\theta)\right)
\end{aligned}$$

We set $f_\theta(x) = \widetilde{\alpha}^{(L)}(x;\theta)$, notice that we specifically use the preactivation to have a final linear combination.

Rescaled vs Classic + LeCun initialization [LeC+12]

Remarks

- initializations of the parameters are different
- $\beta > 0$ is added
- a $\frac{1}{\sqrt{n_\ell}}$ factor is added for each $\ell \in \{0, \ldots, L-1\}$

Rescaled vs Classic + LeCun initialization [LeC+12]

Remarks

- initializations of the parameters are different
- $\beta > 0$ is added
- a $\frac{1}{\sqrt{n_\ell}}$ factor is added for each $\ell \in \{0, \dots, L-1\}$

Scaling is instrumental to observe the asymptotic regime and:

- same representable space $F^{(L)}(\mathbb{R}^P)$
- derivatives $\partial_{W_{ij}^\ell} F^{(L)}, \partial_{b_j^\ell} F^{(L)}$ are scaled by a factor of $\frac{1}{\sqrt{n_\ell}}, \beta$ respectively
- β added to *balance* [JGH20](Remark 1)

Toy ANN

We provide a simpler example for the sake of understanding. Consider an $L = 2$ layer ANN with $n_L = 1$ (i.e. one hidden layer, scalar output).

- 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. All normalized.
- in the second layer parameters are $\{b_j\}_{j=1}^{n_1}$ for each neuron, with b_0 the added bias

Toy ANN

We provide a simpler example for the sake of understanding. Consider an $L = 2$ layer ANN with $n_L = 1$ (i.e. one hidden layer, scalar output).

- 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. All normalized.
- in the second layer parameters are $\{b_j\}_{j=1}^{n_1}$ for each neuron, with b_0 the added bias

The output can be written as:

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

Toy ANN, graphically

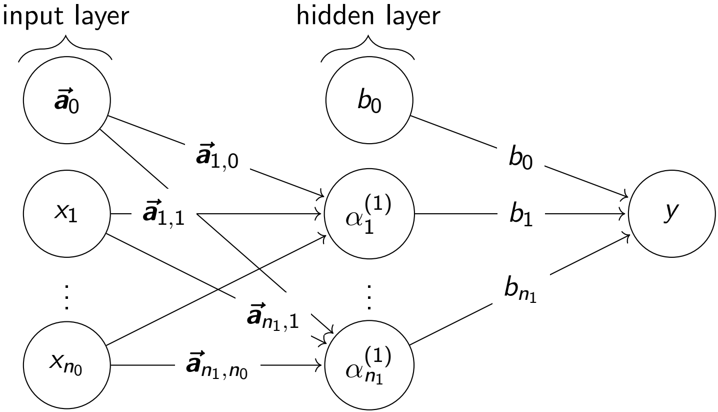

Figure: Activations $\alpha_i^{(1)}$ are $\alpha_i^{(1)}(x) = \sigma(\widetilde{\alpha}_i^{(1)})$. With the architecture considered, , $\vec{a}_0 = \beta\vec{\mathbf{1}}, \beta_0 = \beta$ and $\widetilde{\alpha}^{(1)}, \widetilde{\alpha}^{(2)}$ have the scaling factors $\frac{1}{\sqrt{n_\ell}}$ inside.

Why and what in one slide

Using the result that ANNs 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

Why and what in one slide

Using the result that ANNs 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
- show that the network function obeys a Neural Tangent Kernel Gradient flow with respect to the functional cost (evolves according to a kernel)

Why and what in one slide

Using the result that ANNs 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
- show that the network function obeys a Neural Tangent Kernel Gradient flow with respect to the functional cost (evolves according to a kernel)
- such Kernel is random at initialization and varies, but at the limit and under precise assumptions it is **static**

Recap

We consider the classical fully-connected ANN architecture, rescaled, from a different point of view.

Recap

We consider the classical fully-connected ANN architecture, rescaled, from a different point of view.

- need to understand how kernels enter the discussion in [JGH20]
- will show an interesting application of this to justify a heuristic method

Lecture Path

A partial empirical motivation [Soh20]

Lazy training: as the number of hidden neurons increases, weights are **almost static**.

Remark

This does not mean that we do not learn or that we do not optimize, but just that optimality is *close*.

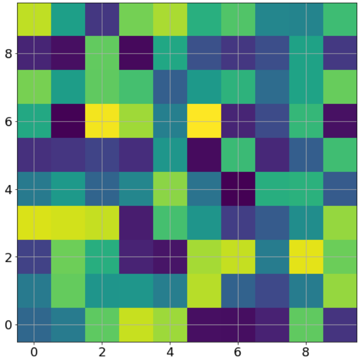

Figure: Small size weight matrix. Source [Vad19]

Many neurons weight matrix dynamics

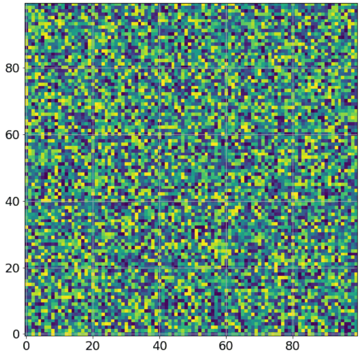

Figure: Medium size. Source [Vad19]

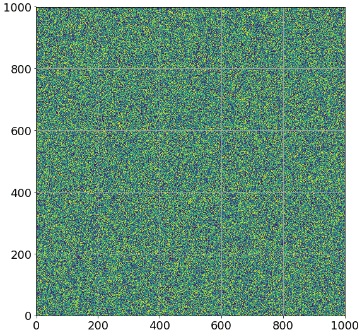

Figure: Big size. Source [Vad19]

Taylor expansion

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 the function is affine in θ or in $\Delta(\theta) = \theta - \theta(0)$.

Remark

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.

Null intercept

Assume $f_{\theta(0)}(x) = 0$. There is a justification for this in [Ten22a].

Linearization

Linearized Model g_θ

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

We can then interpret the expression $\widehat{y} = \left\langle \partial_\theta f_{\theta(0)}(x), \Delta\theta \right\rangle$ as a feature map with kernel:

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

Interpretation

If $\partial_\theta f_{\theta(0)}(x) = \varphi(x)$ then:

- ☺ the expansion looks like **gradient descent**
- ☺ of a **linear model**
- ☺ on a **functional space** with **convex cost**

Recap

We started from an empirical observation and found an object.

Validity

- How reliable is this approximation?
- When is it reliable?
- What is it? (i.e. is there a theoretical approach to put in perspective?)

Recap

We started from an empirical observation and found an object.

Validity

- How reliable is this approximation?
- When is it reliable?
- What is it? (i.e. is there a theoretical approach to put in perspective?)

We will answer all of these.

Lecture Path

The general formulation from Theory

Neural Tangent Kernel, NTK

When the dynamics are $\partial_t f_{\theta(t)} = -\nabla_{\mathbf{\Theta}^{(L)}} C \Big|_{f_{\theta(t)}}$ we say that the NTK is:

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

For elements $x, x' \in \mathscr{X}$ an entry has form
$\mathbf{\Theta}^{(\mathbf{L})}_{\mathbf{ij}}(\theta)(x, x') = \sum_{p=1}^{P} \left[\partial_{\theta_p} F^{(L)}(\theta, x)\right]_i \left[\partial_{\theta_p} F^{(L)}(\theta, y)\right]_j$

The general formulation from Theory

Neural Tangent Kernel, NTK

When the dynamics are $\partial_t f_{\theta(t)} = -\nabla_{\mathbf{\Theta}^{(L)}} C\Big|_{f_{\theta(t)}}$ we say that the NTK is:

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

For elements $x, x' \in \mathscr{X}$ an entry has form
$\mathbf{\Theta}_{\boldsymbol{ij}}^{(\boldsymbol{L})}(\theta)(x, x') = \sum_{p=1}^{P} \left[\partial_{\theta_p} F^{(L)}(\theta, x)\right]_i \left[\partial_{\theta_p} F^{(L)}(\theta, y)\right]_j$

Remark

Actual NTK is **random at initialization and varies during training**! Not the constant at $\partial_\theta f_{\theta(0)}$ as before.

Hypotheses and techniques

Meta-Assumptions

- Sequential layer divergence:

$$\lim_{n_{L-1}\to\infty} \cdots \lim_{n_1\to\infty}$$

- empirical distribution inner product space
- non-linearities are twice differentiable, Lipschitz and with bounded second derivative

Proof Strategy.

- the main strategy is induction on the number of Layers L
- ultimately finding bounds and analysis of the network functions which are Gaussian Processes
- f_θ network functions behavior is the objective
- we avoid **lots of details**

◇

Results I

Network functions are Gaussian Processes

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 with null mean and covariance defined recursively in L by the equations:

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

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

Results II

Kernel Convergence at Initialization

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

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

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

The form of $\boldsymbol{\Theta}_{\infty}^{(L)}$ is described recursively as:

$$\boldsymbol{\Theta}_{\infty}^{(1)}(x, x') = \Sigma^{(1)}(x, x')$$

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

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

Results III

Kernel Convergence across dynamics

it holds that for any T satisfying $\int_0^T \|d_t\|_{p^{in}} dt < \infty$ stochastically:

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

where the symbol $\underset{t\in[0,T]}{\overset{\{n_\ell\}\to\infty}{\rightrightarrows}}$ means in the sequential limit of the hidden neurons uniformly in $t \in [0, T]$.

Then, the network function follows the **Kernel Gradient** [JGH20](Sec. 3) differential equation:

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

Interpretation

Independence at infinite-width limit

Neurons separately converge ($\otimes$). Training an ANN for n_L outputs is equal to training n_L scalar ANNs

Interpretation

Independence at infinite-width limit

Neurons separately converge ($\otimes$). Training an ANN for n_L outputs is equal to training n_L scalar ANNs

Limiting Kernel form

Described by the non-linearity σ, the depth L and the variance of the initialization

Interpretation

Independence at infinite-width limit

Neurons separately converge ($\otimes$). Training an ANN for n_L outputs is equal to training n_L scalar ANNs

Limiting Kernel form

Described by the non-linearity σ, the depth L and the variance of the initialization

During training

The evolution across time of the kernel at the diverging limit is described by a single constant kernel. The *precision* of this convergence is independent of t.

Dynamics Convergence

Remark

The NTK governs the dynamics at infinite-width. Even if it is well-behaved, convergence is not guaranteed, as it might not be positive definite (i.e. null at some point, stuck dynamics before optimality).

Spherical Data NTK

Assume further that σ is **nonpolynomial**. Then, for $L \geq 2$ the restriction to the sphere $\mathbb{S}^{n_0-1}$ of the limiting NTK $\Theta_\infty^{(L)}$ derived before is positive definite, and the dynamics **never stop until convergence**.

Dynamics Convergence

Remark

The NTK governs the dynamics at infinite-width. Even if it is well-behaved, convergence is not guaranteed, as it might not be positive definite (i.e. null at some point, stuck dynamics before optimality).

Spherical Data NTK

Assume further that σ is **nonpolynomial**. Then, for $L \geq 2$ the restriction to the sphere $\mathbb{S}^{n_0-1}$ of the limiting NTK $\boldsymbol{\Theta}_{\infty}^{(L)}$ derived before is positive definite, and the dynamics **never stop until convergence**.

Remark

Data supported on a sphere is a *good* approximation of high-dimensional data [JGH20](App. A.4).

Idea

Assume we can use all the theorems, we have:

- a static deterministic kernel which depends only on:
  - L
  - σ
  - the starting variance $\Sigma^{(1)}$
- also positive definite, guaranteeing convergence to the optimal point

Idea

Assume we can use all the theorems, we have:

- a static deterministic kernel which depends only on:
  - L
  - σ
  - the starting variance $\Sigma^{(1)}$
- also positive definite, guaranteeing convergence to the optimal point

Then, we can split the dynamics into *eigendirections*.

Remark

We will see a simplified version on the $L = 2$ network, not the general case.

NTK quadratic regression cost, toy model

Toy NN update equations

Consider a quadratic loss in the simple setting of $L = 2, n_L = 1$. Mathematically:

$$\mathscr{L}(\theta) = \frac{1}{2}\left\|\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right\|^2 \quad \begin{cases} \partial_\theta \mathscr{L}(\theta) = \left(\partial_\theta \widehat{\vec{\boldsymbol{y}}}\right)^T \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right) \\ \partial_t \theta(t) = -\left(\partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}}\right)^T \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right) \end{cases}$$

In the parameter space at the infinite-width limit the output evolves as:

$$\partial_t \widehat{\vec{\boldsymbol{y}}} = -\left\|\partial_{\theta(t)} \widehat{\vec{\boldsymbol{y}}}\right\|^2 \left(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}\right) \approx -\mathbf{K}(\theta(0))(\widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}})$$

where $\mathbf{K}(\theta(0))$ is the NTK, a **good** approximation.

Infinite-width onvergence

Exponential eigendirection dynamics

Now define $\vec{\boldsymbol{u}} = \widehat{\vec{\boldsymbol{y}}} - \vec{\boldsymbol{y}}$ and see that:

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

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, no multiple eigenvalues:

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

Exponential convergence has rate $\min\{\lambda_i\} = \lambda_1$.

Early stopping Heuristics

Briefly:

- dynamics separated along the eigenspaces
- the *speed of convergence* is different and governed by λ_i
- the bigger the variation inside the eigenspace, the faster the convergence
- to a low variation (eigenvalue) we associate noise

Early stopping Heuristics

Briefly:

- dynamics separated along the eigenspaces
- the *speed of convergence* is different and governed by λ_i
- the bigger the variation inside the eigenspace, the faster the convergence
- to a low variation (eigenvalue) we associate noise

Early Stopping justification

Let the learning flow until **not all of the directions** have saturated. By **stopping early**, low variation directions have not converged.

Empirical Results on General Model

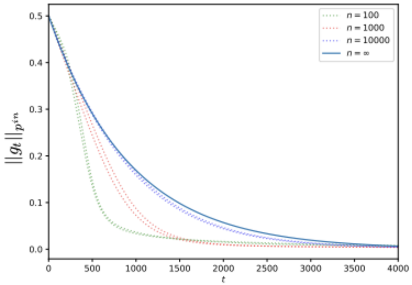

Figure: Norm dynamics over time, parallel direction

g_θ plot, n are the sizes of hidden neurons. As n increases, approaches exponential hypothesis.

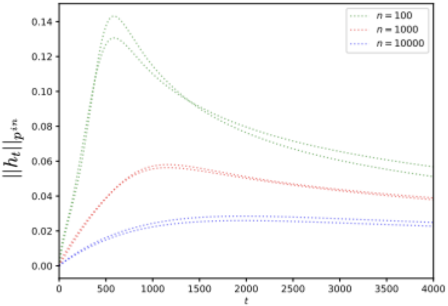

Figure: Norm dynamics over time, orthogonal direction

h_θ plot. n are the sizes of hidden neurons. As n increases, approaches null hypothesis

Lecture Path

Recap

Results in [JGH20] make use of:

- Kernel Methods
- Dual vector spaces
- thougthful general problem construction

Recap

Results in [JGH20] make use of:

- Kernel Methods
- Dual vector spaces
- thougthful general problem construction

to show:

- that ANNs at the infinite-width limit behave like Kernels
- good experimental results
- that the framework has other intepretations (see [JGH20])

Recap

Results in [JGH20] make use of:

- Kernel Methods
- Dual vector spaces
- thougthful general problem construction

to show:

- that ANNs at the infinite-width limit behave like Kernels
- good experimental results
- that the framework has other intepretations (see [JGH20])

Pros

- ☺ gradient descent/flow
- ☺ theoretical results
- ☺ **reasonable** assumptions

Recap

Weaknesses

- ☹ ANNs
- ☹ does not match SOTA
- ☹ only a partial description of DL architectures

Additional/important refs:

- No sequential limit result and NTK for CNNs [Aro+19]
- Kernel methods theory [SC04]
- Code implementations [Aro+22], or Papers with Code NTK page
- further details about NTKs [COB20]

Concluding

Any question/discussion, let me know!

Thank you!

simonegiancola09@gmail.com

personal webpage

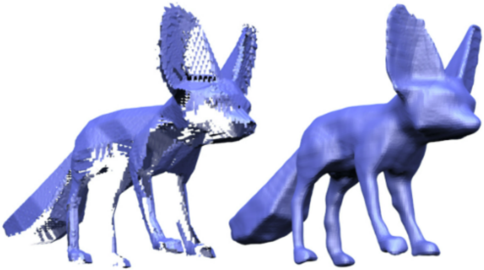

Figure: NTK reconstructed fox. Source [CPW21]

References I

[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].

[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.

References II

[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.

[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.

References III

[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.

References IV

[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].

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

[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.

References V

[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.

[COB20] Lenaic Chizat, Edouard Oyallon, and Francis Bach. *On Lazy Training in Differentiable Programming*. Jan. 2020. DOI: 10.48550/arXiv.1812.07956. arXiv: 1812.07956 [cs, math].

[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].