Artificial Neural Networks and Kernel Methods

Franck Gabriel, Joint works with Arthur Jacot, Clément Hongler,
François Ged, Berfin Şimşek, Francesco Spadaro.
Chair of Statistical Field Theory, EPFL

DataSig Seminar

23th July 2020

Introduction

Two competing methods in machine learning: **Neural Networks** and **Kernel Methods.**

Question: Did N.N. end the game ? Or is it a never-ending war ? Can these methods interact with each other ?

Introduction

Two competing methods in machine learning: **Neural Networks** and **Kernel Methods.**

Question: Did N.N. end the game ? Or is it a never-ending war ? Can these methods interact with each other ?

Outline of the talk:

- ▷ **Introduction to Supervised Learning.**
- ▷ **Neural Networks and Neural Tangent Kernel.**
- ▷ **Theoretical and Practical Consequences.**
- ▷ **Extreme Learning and Regularized Kernel Methods.**
- ▷ **Kernel Method Generalization from the training set.**

Answer: Deep connections and interplay between Neural Networks and Kernel Methods.

Introduction to Supervised Learning

Abstraction and the Four Main Questions

Ideal Goal: to predict
e.g. : age of a person in a picture
$\downarrow$
Realistic Goal : train data
e.g. : "few" labelled pictures

Parameterized family
of functions : $(f_\theta)_{\theta \in \mathbb{R}^P}$

training part
$\longleftarrow$
$\longrightarrow$
optimization : θ^*

$\nearrow$
$\searrow$

Existence is not enough,
we want to find it.

$\downarrow$
Generalization
You hope that the ideal goal is achieved

Abstraction and the Four Main Questions

Ideal Goal: to predict
e.g. : age of a person in a picture
$\downarrow$
Realistic Goal : train data
e.g. : "few" labelled pictures

Parameterized family
of functions : $(f_\theta)_{\theta \in \mathbb{R}^P}$

training part
$\longleftarrow$
$\longrightarrow$
optimization : θ^*

$\nearrow$
$\searrow$

Existence is not enough,
we want to find it.

$\downarrow$

Generalization
You hope that the ideal goal is achieved

Does it learn ?	How does it learn ?	What does it learn ?	Is it useful ?
Training error	$(f_{\theta_t})_{t \geq 0}$	f_{θ^*}	Generalization error

General Setup: Regression, Predict $f^* : \mathbb{R}^{n_0} \longrightarrow \mathbb{R}^{n_{out}}$

Always assume $n_{out} = 1$, generalizable to $n_{out} > 1$.

Goal:

▷ *Ideal:* $\forall x, f_\theta(x) \sim f^*(x)$,

▷ *Proxy:* Functional Cost, e.g. M.S.E

$$\mathcal{C}(f) = \frac{1}{2} \int (f(x) - f^*(x))^2 \, d\mu(x),$$

▷ *Dataset:* $(x_i, y_i := f^*(x_i))_{i=1,\ldots,N}$,

▷ *Cost function* : Cost $\sim 0 \iff$ Goal achieved, e.g.

$$\mathcal{C}_N(f) = \frac{1}{2N} \sum_{i=1}^{N} (f(x_i) - y_i)^2 ,$$

General Setup: Regression, Predict $f^* : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{out}}$

Always assume $n_{out} = 1$, generalizable to $n_{out} > 1$.

Goal:

▷ *Ideal:* $\forall x, f_\theta(x) \sim f^*(x)$,

▷ *Proxy:* Functional Cost, e.g. M.S.E

$$\mathcal{C}(f) = \frac{1}{2} \int (f(x) - f^*(x))^2 \, d\mu(x),$$

▷ *Dataset:* $(x_i, y_i := f^*(x_i))_{i=1,\dots,N}$,

▷ *Cost function* : Cost $\sim 0 \iff$ Goal achieved, e.g.

$$\mathcal{C}_N(f) = \frac{1}{2N} \sum_{i=1}^{N} (f(x_i) - y_i)^2,$$

Model:

▷ *Parameterization:*

$$F : \theta \in \mathbb{R}^P \to \mathcal{F},$$

▷ *Parameters Cost Function:*

$$C = \mathcal{C}_N \circ F,$$

$\mathcal{C}_N$ is often convex, F can be not linear
$\Rightarrow$ the cost C might be non convex.

General Setup: Regression, Predict $f^* : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{out}}$

Always assume $n_{out} = 1$, generalizable to $n_{out} > 1$.

Goal:

▷ *Ideal:* $\forall x, f_\theta(x) \sim f^*(x)$,

▷ *Proxy:* Functional Cost, e.g. M.S.E

$$\mathcal{C}(f) = \frac{1}{2} \int (f(x) - f^*(x))^2 \, d\mu(x),$$

▷ *Dataset:* $(x_i, y_i := f^*(x_i))_{i=1,\dots,N}$,

▷ *Cost function* : Cost $\sim 0 \iff$ Goal achieved, e.g.

$$\mathcal{C}_N(f) = \frac{1}{2N} \sum_{i=1}^{N} (f(x_i) - y_i)^2,$$

Model:

▷ *Parameterization:*

$$F : \theta \in \mathbb{R}^P \to \mathcal{F},$$

▷ *Parameters Cost Function:*

$$C = \mathcal{C}_N \circ F,$$

$\mathcal{C}_N$ is often convex, F can be not linear
$\Rightarrow$ the cost C might be non convex.

Problem : Minimize C with an explicit algorithm: $\arg\min_\theta C(\theta)$.

Motivation: Two competing spaces of functions

Kernel methods

▷ $(\mathcal{H}, \langle\rangle_{\mathcal{H}})$ *Hilbert space* of real valued functions, evaluation on x continuous:

$$f(x) = \langle f, K_x \rangle_{\mathcal{H}} .$$

The kernel $K(x, y) = K_x(y)$ satisfies:

1. Symmetric $K(x, y) = K(y, x)$,
2. Matrices $(K(x_i, x_j))_{i,j}$ are positive semidefinite.

▷ Find f^* minimal norm in $\mathcal{H}$ such that $f(x_i) = y_i$ (or MSE+$\lambda \|f\|^2_{\mathcal{H}}$, $\lambda \searrow 0$).

▷ *Representer theorem*: f^* of the form

$$f_\theta(\cdot) = \sum_{i=1}^{N} \theta_i K(x_i, \cdot).$$

▷ Solution: $\theta^* = K(X, X)^{-1} Y$.

▷ **Ridgeless Kernel Regression:**

$$f_{\theta^*}(\cdot) = \sum_{i=1}^{N} \theta_i^* K(x_i, \cdot).$$

Motivation: Two competing spaces of functions

Kernel methods

▷ $(\mathcal{H}, \langle\rangle_{\mathcal{H}})$ *Hilbert space* of real valued functions, evaluation on x continuous:

$$f(x) = \langle f, K_x \rangle_{\mathcal{H}} .$$

The kernel $K(x, y) = K_x(y)$ satisfies:

1. Symmetric $K(x, y) = K(y, x)$,
2. Matrices $(K(x_i, x_j))_{i,j}$ are positive semidefinite.

▷ Find f^* minimal norm in $\mathcal{H}$ such that $f(x_i) = y_i$ (or MSE+$\lambda \|f\|^2_{\mathcal{H}}, \lambda \searrow 0$).

▷ *Representer theorem*: f^* of the form

$$f_\theta(\cdot) = \sum_{i=1}^{N} \theta_i K(x_i, \cdot).$$

▷ Solution: $\theta^* = K(X, X)^{-1} Y$.

▷ **Ridgeless Kernel Regression:**

$$f_{\theta^*}(\cdot) = \sum_{i=1}^{N} \theta_i^* K(x_i, \cdot).$$

Fully connected Artificial Neural Networks

▷ A parameterization of a *dense space of* functions:

$$f_\theta : \mathbb{R}^{n_0} \underset{A_1}{\longrightarrow} \mathbb{R}^{n_1} \underset{\sigma}{\longrightarrow} \mathbb{R}^{n_1} \underset{A_2}{\longrightarrow} \mathbb{R}^{n_2} \underset{\sigma}{\longrightarrow} \mathbb{R}^{n_2} \longrightarrow \ldots \mathbb{R}^{n_{L-1}} \underset{\sigma}{\longrightarrow} \mathbb{R}^{n_{L-1}} \underset{A_L}{\longrightarrow} \mathbb{R}^{n_{out}}$$

with:

1. $A_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ an affine function (the parameters),
2. $\boldsymbol{\sigma}$ the pointwise application of a non-linearity $\sigma : \mathbb{R} \to \mathbb{R}$.

▷ Find θ^* which minimizes the cost C.

▷ **Gradient descent.**

▷ Beliefs : Gradient descent will be stuck in good minimum.

Questions and answers

Are they so different?

1. Infinite Width Neural Network = Kernel Method
2. Infinite Width Neural Network with finite last hidden layer $\sim$ Kernel Method with Regularization

Can Kernel Method Theory give us a better insight on A.N.N.?

1. It allows us to answer the Four Main Questions for Infinite Width Neural Network: does it learn ? How does it learn ? What does it learn ? Does it generalize ?
2. Better insight into the architectural design of A.N.Ns.

Neural Networks and Neural Tangent Kernel

Main result: take away

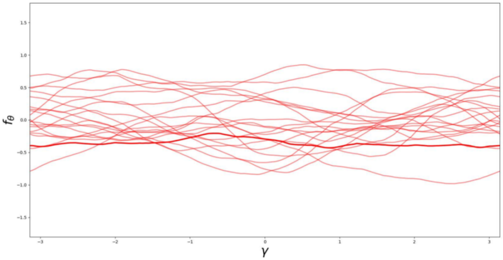
Theorem (Jacot, Gabriel, Hongler, NeuRIPS 2018)

Gradient Descent Learning for Infinite Width Limit Neural Networks

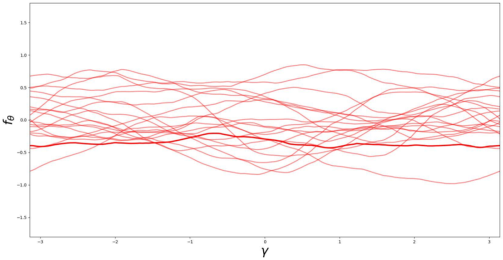
$\|$

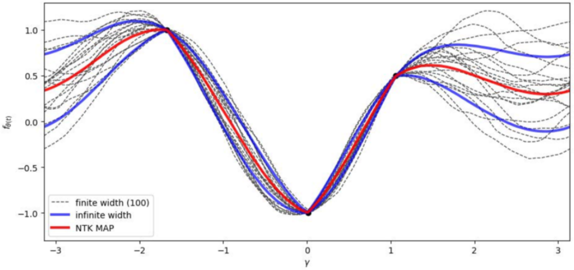
Kernel Method for the Neural Tangent Kernel (N.T.K.)

Illustration



Illustration

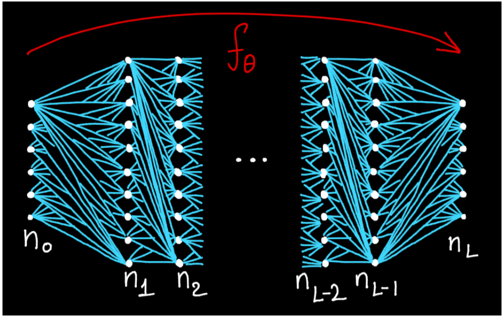




Setup: Fully Connected Neural Networks

A Fully Connected Neural Network:

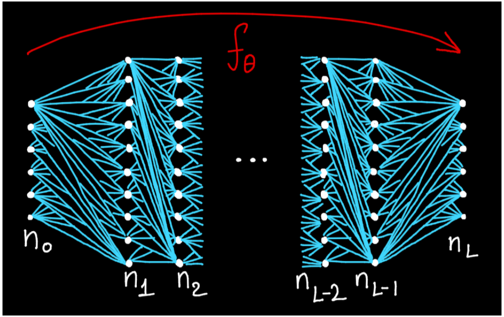
- **Non linearity:** $\sigma : \mathbb{R} \to \mathbb{R}$, e.g. $ReLU(x) = \max(0, x)$. (Lipschitz, twice differentiable nonlinearity function for our theorem),
- **Number of hidden layers:** $L - 1$,
- **Sizes of the layers:** $n_{in} = n_0, n_1, \ldots, n_{L-1}, n_L = n_{out} = 1$.



Setup: Fully Connected Neural Networks

A Fully Connected Neural Network:

- **Non linearity:** $\sigma : \mathbb{R} \to \mathbb{R}$, e.g. $ReLU(x) = \max(0, x)$. (Lipschitz, twice differentiable nonlinearity function for our theorem),
- **Number of hidden layers:** $L - 1$,
- **Sizes of the layers:** $n_{in} = n_0, n_1, \ldots, n_{L-1}, n_L = n_{out} = 1$.



$$f_\theta^{(L)} : \mathbb{R}^{n_0} \xrightarrow[x \mapsto \frac{1}{\sqrt{n_0}} W^{(0)} x + \beta b^{(0)}]{} \mathbb{R}^{n_1} \xrightarrow{\sigma} \mathbb{R}^{n_1} \xrightarrow[x \mapsto \frac{1}{\sqrt{n_1}} W^{(1)} x + \beta b^{(1)}]{} \ldots \xrightarrow{\sigma} \mathbb{R}^{n_{L-1}} \xrightarrow[x \mapsto \frac{1}{\sqrt{n_{L-1}}} W^{(L-1)} x + \beta b^{(L-1)}]{} \mathbb{R}$$

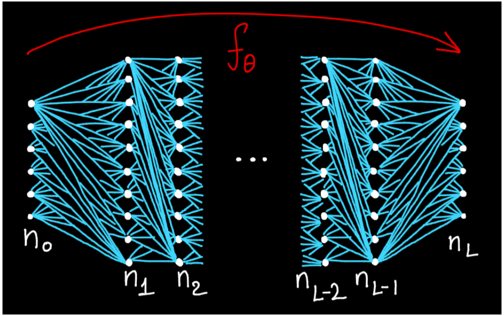
▷Ptw. application of σ,

▷The parameters : $(\theta_p)_{p \in [P]} = \left(W^{(0)}, b^{(0)}, \ldots, W^{(L-1)}, b^{(L-1)}\right)$.

Setup: Fully Connected Neural Networks

A Fully Connected Neural Network:

- **Non linearity:** $\sigma : \mathbb{R} \to \mathbb{R}$, e.g. $ReLu(x) = x \vee 0$. (Lipschitz, twice differentiable nonlinearity function for our theorem),
- **Number of hidden layers:** $L - 1$,
- **Size of the layers:** $n_{in} = n_0, n_1, \ldots, n_{L-1}, n_L = n_{out} = 1$.



Activations $\alpha^{(\ell)}$. Preactivations $\tilde{\alpha}^{(\ell)}$. Output function $f_\theta(x) = \tilde{\alpha}^{(L)}(x)$

$$\tilde{\alpha}^{(\ell+1)}(x) = \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \alpha^{(\ell)}(x) + \beta b^{(\ell)},$$

$$\alpha^{(\ell+1)}(x) = \sigma\left(\tilde{\alpha}^{(\ell+1)}(x)\right),$$

with pointwise application of σ.

Setup: Algorithm, the gradient descent

We implement a *first-order algorithm* and we want the cost to decrease:

$$\theta \to \theta + d\theta \quad \Rightarrow \quad C(\theta) \to C(\theta) + \langle \nabla C(\theta), d\theta \rangle$$

$$\hookrightarrow \quad d\theta \propto -\nabla C(\theta)$$

Setup: Algorithm, the gradient descent

We implement a *first-order algorithm* and we want the cost to decrease:

$$\theta \to \theta + d\theta \quad \Rightarrow \quad C(\theta) \to C(\theta) + \langle \nabla C(\theta), d\theta \rangle$$

$$\hookrightarrow \qquad d\theta \propto -\nabla C(\theta)$$

Cost

$C = \mathcal{C}_N \circ F$, i.e.

$$C(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$$

Algorithm

Gradient Descent:

$$d\theta = -\nabla C(\theta) dt,$$

Gradient Flow:

$$\partial_t \theta_t = -\nabla C(\theta_t)$$

Initialization

If $(\theta_p)_{p=1,\ldots P} = 0$, the gradient descent gets stuck.
Idea [LeCun/He init.]

$$(\theta_p)_{p=1,\ldots P} \sim \mathcal{N}(0,1) \text{ i.i.d.}$$

New Object: The N.T.K.

How can we describe the training of N.N?

Study the dynamics of f_{θ_t} and not of θ_t.

How?

Using a new kernel

New Object: The N.T.K.

How can we describe the training of N.N?

Study the dynamics of f_{θ_t} and not of θ_t.

How?

Using a new kernel

The Neural Tangent Kernel

$$\Theta^{(L)}(x_1, x_2) = \sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x_1) \frac{\partial f_\theta}{\partial \theta_p}(x_2) = \langle \nabla_\theta f_\theta(x_1), \nabla_\theta f_\theta(x_2) \rangle .$$

New Object: The N.T.K.

How can we describe the training of N.N?

Study the dynamics of f_{θ_t} and not of θ_t.

How?

Using a new kernel

The Neural Tangent Kernel

$$\Theta^{(L)}(x_1, x_2) = \sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x_1) \frac{\partial f_\theta}{\partial \theta_p}(x_2) = \langle \nabla_\theta f_\theta(x_1), \nabla_\theta f_\theta(x_2) \rangle \, .$$

It is *random* at initialization and *evolves with time.*

NTK and the learning dynamics.

The Neural Tangent Kernel

$$\Theta^{(L)}(x_1, x_2) = \sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x_1) \frac{\partial f_\theta}{\partial \theta_p}(x_2) = \langle \nabla_\theta f_\theta(x_1), \nabla_\theta f_\theta(x_2) \rangle .$$

Theorem (Jacot, Gabriel, Hongler 18)

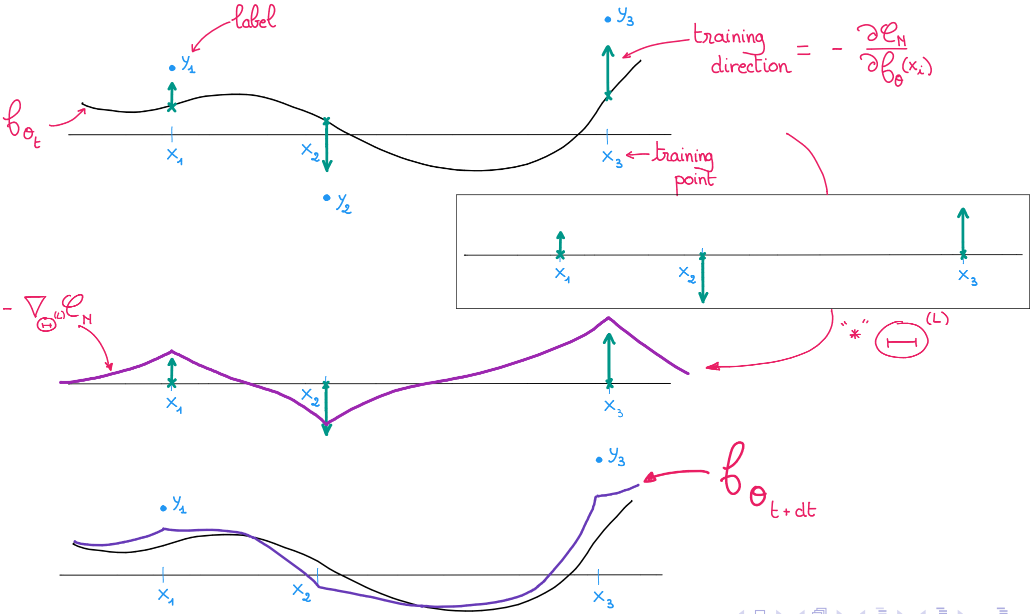
Consider a Fully Connected Neural Network with $L-1$ hidden layers of width $n_1, \ldots n_{L-1}$: $f_\theta : \mathbb{R}^{n_{in}} \to \mathbb{R}$. During Gradient Descent:

$$\partial_t f_{\theta_t} = -\nabla_{\Theta_t^{(L)}} \mathcal{C}_N,$$

where

$$\nabla_{\Theta_t^{(L)}} \mathcal{C}_N(x) = \sum_{i=1}^{N} \Theta_t^{(L)}(x, x_i) \frac{\partial \mathcal{C}_N}{\partial f_{\theta_t}(x_i)}.$$

Illustration: Dynamics



Proof: Dynamics

Recall that $C = \mathcal{C}_N \circ F$, with $\mathcal{C}_N(f) = c(f(x_1), \ldots, f(x_N))$.

- Parameter Space: $d\theta_p = -\frac{\partial C}{\partial \theta_p} dt = -\sum_{i=1}^{N} \frac{\partial f_\theta}{\partial \theta_p}(x_i) \frac{\partial \mathcal{C}_N}{\partial y_i} dt$.

Proof: Dynamics

Recall that $C = \mathcal{C}_N \circ F$, with $\mathcal{C}_N(f) = c(f(x_1), \dots, f(x_N))$.

- Parameter Space: $d\theta_p = -\frac{\partial C}{\partial \theta_p} dt = -\sum_{i=1}^{N} \frac{\partial f_\theta}{\partial \theta_p}(x_i) \frac{\partial \mathcal{C}_N}{\partial y_i} dt$.
- Function Space:

$$f_\theta(x) \to f_{\theta + d\theta}(x) \quad \sim \quad f_\theta(x) + \sum_{p=1}^{P} d\theta_p \frac{\partial f_\theta}{\partial \theta_p}(x)$$

$$f_\theta(x) - \sum_{i=1}^{N} \left[\sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x) \frac{\partial f_\theta}{\partial \theta_p}(x_i) \right] \frac{\partial \mathcal{C}_N}{\partial f(x_i)} dt.$$

Proof: Dynamics

Recall that $C = \mathcal{C}_N \circ F$, with $\mathcal{C}_N(f) = c(f(x_1), \ldots, f(x_N))$.

- Parameter Space: $d\theta_p = -\frac{\partial C}{\partial \theta_p} dt = -\sum_{i=1}^{N} \frac{\partial f_\theta}{\partial \theta_p}(x_i) \frac{\partial \mathcal{C}_N}{\partial y_i} dt$.
- Function Space:

$$
\begin{aligned}
f_\theta(x) \to f_{\theta+d\theta}(x) \quad &\sim \quad f_\theta(x) + \sum_{p=1}^{P} d\theta_p \frac{\partial f_\theta}{\partial \theta_p}(x) \\
& \quad\; f_\theta(x) - \sum_{i=1}^{N} \left[\sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x) \frac{\partial f_\theta}{\partial \theta_p}(x_i) \right] \frac{\partial \mathcal{C}_N}{\partial f(x_i)} dt.
\end{aligned}
$$

- Neural Tangent Kernel: $\Theta^{(L)}(x, x_i) = \sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x) \frac{\partial f_\theta}{\partial \theta_p}(x_i)$.

Proof: Dynamics

Recall that $C = \mathcal{C}_N \circ F$, with $\mathcal{C}_N(f) = c(f(x_1), \ldots, f(x_N))$.

- Parameter Space: $d\theta_p = -\frac{\partial C}{\partial \theta_p} dt = -\sum_{i=1}^{N} \frac{\partial f_\theta}{\partial \theta_p}(x_i) \frac{\partial \mathcal{C}_N}{\partial y_i} dt$.
- Function Space:

$$
\begin{aligned}
f_\theta(x) \to f_{\theta+d\theta}(x) \quad \sim \quad & f_\theta(x) + \sum_{p=1}^{P} d\theta_p \frac{\partial f_\theta}{\partial \theta_p}(x) \\
& f_\theta(x) - \sum_{i=1}^{N} \left[\sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x) \frac{\partial f_\theta}{\partial \theta_p}(x_i) \right] \frac{\partial \mathcal{C}_N}{\partial f(x_i)} dt.
\end{aligned}
$$

- Neural Tangent Kernel: $\Theta^{(L)}(x, x_i) = \sum_{p=1}^{P} \frac{\partial f_\theta}{\partial \theta_p}(x) \frac{\partial f_\theta}{\partial \theta_p}(x_i)$.
- Dynamics:

$$
\partial_t f_{\theta_t}(x) = -\sum_{i=1}^{N} \Theta^{(L)}(x, x_i) \frac{\partial \mathcal{C}_N}{\partial y_i} dt = -\nabla_{\Theta^{(L)}} \mathcal{C}_N.
$$

Main Theorem

Theorem (Jacot, Gabriel, Hongler 18)

Consider a Fully Connected Neural Network with $L-1$ hidden layers of width $n_1, \ldots n_{L-1}$: $f_\theta : \mathbb{R}^{n_{in}} \to \mathbb{R}$.

1. *During Gradient Descent:*
$$\partial_t f_{\theta_t} = -\nabla_{\Theta_t^{(L)}} \mathcal{C}_N.$$
2. *When $n_1, \ldots, n_{L-1} \to \infty$ sequentially:*
   - *At initialization, $f_{\theta_0} \sim \mathcal{N}(0, \Sigma^{(L)})$ [Neal 96, de G. Matthews and al 17,18].*
   - *The NTK:*
     - *At initialization, becomes **deterministic**:*
$$\Theta_{t=0}^{(L)} \longrightarrow \Theta_{t=0,\infty}^{(L)}.$$
     - *Becomes **fixed during training**: uniformly on $t \leq T$*
$$\left|\Theta_t^{(L)}(x_1, x_2) - \Theta_{t=0,\infty}^{(L)}(x_1, x_2)\right| \to 0.$$

Limiting dynamics

The limiting trajectory is

$$\partial_t f_{\theta_t} = -\nabla_{\Theta_\infty^{(L)}} \mathcal{C},$$

which **converges to a global minimum** if the cost functional $\mathcal{C}$ is convex and lower bounded and $\Theta_\infty^{(L)}$ is positive definite.

Limiting dynamics

The limiting trajectory is

$$\partial_t f_{\theta_t} = -\nabla_{\Theta_\infty^{(L)}} \mathcal{C},$$

which **converges to a global minimum** if the cost functional $\mathcal{C}$ is convex and lower bounded and $\Theta_\infty^{(L)}$ is positive definite.

Theorem (Jacot, Gabriel, Hongler 18)

Assume that the data $x_1, \dots, x_N$ lie on a sphere:
$\Theta_\infty^{(L)}$ is definite positive for any input dimension n_{in} i.i.f. σ is a non polynomial function.

General Idea

Main Idea: break down an FCNN of size $L+1$ as a FCNN of size L followed by the pointwize application of σ and an affine map.

$$f_\theta^{(L+1)} : \mathbb{R}^{n_0} \xrightarrow{f_\theta^{(L)}} \mathbb{R}^{n_L} \xrightarrow{\sigma} \mathbb{R}^{n_L} \xrightarrow{A_L} \mathbb{R}$$

And use the chain rule.

General Idea

Main Idea: break down an FCNN of size $L+1$ as a FCNN of size L followed by the pointwize application of σ and an affine map.

$$f_\theta^{(L+1)} : \mathbb{R}^{n_0} \xrightarrow{f_\theta^{(L)}} \mathbb{R}^{n_L} \xrightarrow{\sigma} \mathbb{R}^{n_L} \xrightarrow{A_L} \mathbb{R}$$

And use the chain rule.

This intuition holds during:

- **inference:** i.e. when you evaluate $f_\theta^{(L+1)}$,
- **training:** training $f_\theta^{(L+1)}$ means training A_L and training $f_\theta^{(L)}$ with a time dependent cost $\mathcal{C}(A_L\sigma(.))$.

General Idea

Main Idea: break down an FCNN of size $L+1$ as a FCNN of size L followed by the pointwize application of σ and an affine map.

$$f_\theta^{(L+1)} : \mathbb{R}^{n_0} \xrightarrow{f_\theta^{(L)}} \mathbb{R}^{n_L} \xrightarrow{\sigma} \mathbb{R}^{n_L} \xrightarrow{A_L} \mathbb{R}$$

And use the chain rule.

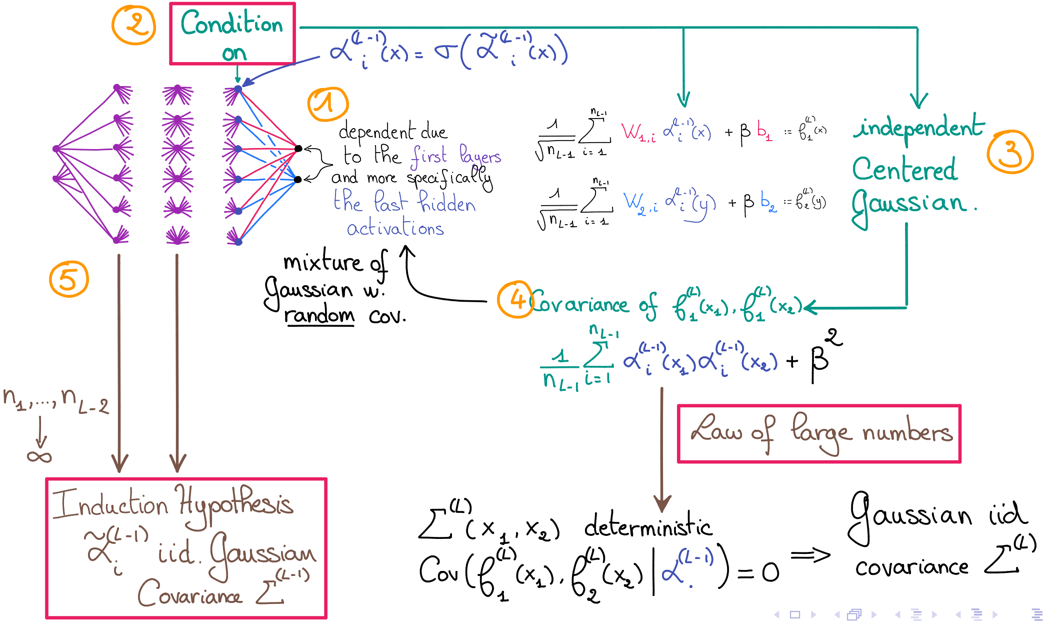
This intuition holds during:

- **inference:** i.e. when you evaluate $f_\theta^{(L+1)}$,
- **training:** training $f_\theta^{(L+1)}$ means training A_L and training $f_\theta^{(L)}$ with a time dependent cost $\mathcal{C}(A_L\sigma(.))$.

Main Tools:

- Induction on the number of layers L,
- Law of large number,
- CLT,
- Generalized Grönwall's inequalities.

Proof: $f_{\theta_L} \sim \mathcal{N}$



Proof: $\Theta^{(L)}_{t,n} \to \Theta^{(L)}_{\infty}$, the inner parameters

$$f_\theta^{(L+1)}(x) = \frac{1}{\sqrt{n_L}} \sum_{k=1}^{n_L} W^{(L)}_{1,k}\, \sigma\left(f_\theta^{(L)}(x)\right) + \beta\, b_1^{(L)}$$

$$\Theta^{(L+1)}_{\text{inner}}(x_1,x_2) = \sum_{\theta_p \in \text{ANN}^{(L)}} \frac{\partial}{\partial \theta_p} f_\theta^{(L+1)}(x_1)\, \frac{\partial}{\partial \theta_p} f_\theta^{(L+1)}(x_2)$$

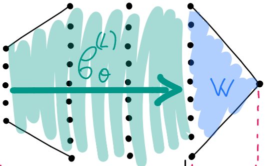
$$\frac{1}{\sqrt{n_L}} \sum_{k=1}^{n_L} W^{(L)}_{1,k}\, \dot\sigma\left(f_\theta^{(L)}(x_1)\right) \frac{\partial}{\partial \theta_p} f_\theta^{(L)}(x_1) \qquad \frac{1}{\sqrt{n_L}} \sum_{k=1}^{n_L} W^{(L)}_{1,k}\, \dot\sigma\left(f_\theta^{(L)}(x_2)\right) \frac{\partial}{\partial \theta_p} f_\theta^{(L)}(x_2)$$

$$\Theta^{(L+1)}_{\text{inner}}(x_1,x_2) = \frac{1}{n_L} \sum_{k,k'=1}^{n_L} W^{(L)}_{1,k} W^{(L)}_{1,k'}\, \dot\sigma\left(f_\theta^{(L)}(x_1)\right) \dot\sigma\left(f_\theta^{(L)}(x_2)\right) \Theta^{(L)}_{k,k'}(x_1,x_2)$$

Law of Large Numbers ⟸ Induction Hyp. $n_1, \dots, n_{L-1} \to \infty$

$\Theta^{(L+1)}_{\text{inner},\infty}(x_1,x_2)$ deterministic $\qquad \delta_{k,k'}\, \Theta^{(L)}_\infty(x_1,x_2)$

Sketch of proof: $\Theta_t^{(L)} \to \Theta_\infty^{(L)}$



(1) train the first layers with a time dep cost: $C_N\left(\frac{1}{\sqrt{n_L}}\sum_{k=1}^{n_L} W_k^{(L)}(t)\,\sigma(f_{\theta_t,k}^{(L)})+\beta b_1\right)$

train the last layer (3)

Which dynamics $n_1,\dots,n_{L-1}\to\infty$? Induction Hyp (2)

$$\partial_t f_\theta^{(L)} = -\frac{1}{\sqrt{n_L}}\,\Theta_\infty^{(L)}\otimes Id\,(\dots)$$ $W_k^{(L)}$

$$d\,W_k(t) = -\frac{1}{\sqrt{n_L}}\sum_{i=1}^{N}\sigma(f_{\theta_t}^{(L)}(x_i))\,\frac{\partial C_N}{\partial y_i}\,dt$$

(4) $f_\theta^{(L)} \xrightarrow{1/\sqrt{n_L}} W_k^{(L)}(t)$

$W_k^{(L)}(t) \xrightarrow{1/\sqrt{n_L}} \Theta_p$ $\Longrightarrow$ Grönwall

$\Theta_p \to f_\theta^{(L)}$: NTK dynamics

W, $f_\theta^{(L)}$ vary at rate $1/\sqrt{n_L}$ $\Longrightarrow$ idem pour NTK

Generalization: multiple output

Generalize to multi-dimensional output:

- $\Theta^{(L)}_{k,k'}(x,x') = \sum_{p=1,\dots,P} \frac{\partial f_{\theta,k}}{\partial \theta_p}(x)\frac{\partial f_{\theta,k'}}{\partial \theta_p}(x')$.
- $\partial_t f_{\theta_t} = -\nabla_{\Theta^{(L)}_t} \mathcal{C}_N$ with $\left(\nabla_{\Theta^{(L)}_t} \mathcal{C}_N\right)_k = \sum_{i=1}^{N} \sum_{k'=1}^{n_{out}} \Theta^{(L)}_{t,k,k'}(\cdot, x_i) \frac{\partial \mathcal{C}_N}{\partial f_{\theta_t,k'}(x_i)}$.

Generalization: multiple output

Generalize to multi-dimensional output:

- $\Theta^{(L)}_{k,k'}(x,x') = \sum_{p=1,\dots,P} \frac{\partial f_{\theta,k}}{\partial \theta_p}(x) \frac{\partial f_{\theta,k'}}{\partial \theta_p}(x')$.
- $\partial_t f_{\theta_t} = -\nabla_{\Theta^{(L)}_t} \mathcal{C}_N$ with $\left(\nabla_{\Theta^{(L)}_t} \mathcal{C}_N\right)_k = \sum_{i=1}^{N} \sum_{k'=1}^{n_{out}} \Theta^{(L)}_{t,k,k'}(\cdot, x_i) \frac{\partial \mathcal{C}_N}{\partial f_{\theta_t,k'}(x_i)}$.

Main Features of the Multiple Output Setting:

- At initialization, $(f_{\theta_0,k})_{k=1}^{n_{out}}$ are **i.i.d.**
- The limiting NTK is **diagonal:**

$$\left(\Theta^{(L)}_\infty\right)_{k,k'}(x,x') = \left(\Theta^{(L)}_\infty(x,x')\right)\delta_{k,k'}.$$

- The functions $(f_{\theta,k})_{k=1}^{n_{out}}$ **evolve independently.**

Other Generalizations

Since then (May 2018), many generalizations:

- Finite time step, large but finite width, infinite time horizon for M.S.E **[Du S. for 2 layers ReLU $\sim$ NTK (2018)], [Allen-Zhu et al. (2018)], [many papers of Arora S., Du S. and al (2019)]**
- Lazy training: **[Chizat-Bach (2018)]**, **[Lee, Xiao and al. (2019)]**
- Taylorised learning **[Huang, Yau (2019)]** : Neural Tangent Hierarchy, N^3 width enough. Fluctuations($\Theta^{(L)}_{t=0}$)$\sim P^{-\frac{1}{4}}$, Fluctuations($\Theta^{(L)}_t - \Theta^{(L)}_{t=0}$)$\sim P^{-\frac{1}{2}}$. **[Bai and al. (2020)]**
- Other architectures at initialization: Tensor Programs of Greg Yang (2019)
- Other optimization algorithm:
  - Momentum **[Lee, Xiao and al. (2019)]**
  - Natural gradient **[Rudner, Teh, Wenzel, Gal (2019)]**

Theoretical and Practical Consequence on ANN Learning

Reminder

The dynamics of f_{θ_t} during training is given by:

Finite size		Randomness	Large limit
	$\nearrow$	Random initial kernel $\Theta^{(L)}_{t=0}$	Deterministic
$\partial_t f_{\theta_t} = -\nabla_{\Theta^{(L)}_t} \mathcal{C}_N,$	$\longrightarrow$	Random evolution of $\Theta^{(L)}_t$	Constant in time
f_{θ_0}	$\searrow$	Random initial function f_{θ_0}	$f_{\theta_0} \sim \mathcal{N}\left(0, \Sigma^{(L)}\right)$

Answer to the Four Questions: how and what?

General setting

- Dynamics:

$$\partial_t f_{\theta_t}(x) = -\sum_{x_i} \Theta^{(L)}(x, x_i) \frac{\partial \mathcal{C}_N}{\partial f_{\theta_t}(x_i)}$$

  Hence: $f_{\theta_t} = f_0 + \sum \vartheta_{i,t} \Theta^{(L)}(x, x_i)$.
- Final function:
  $f_0 +$ Kernel method for $\mathcal{C}_N(\cdot + f_0)$

Answer to the Four Questions: how and what?

General setting

- Dynamics:

$$\partial_t f_{\theta_t}(x) = -\sum_{x_i} \Theta^{(L)}(x, x_i) \frac{\partial \mathcal{C}_N}{\partial f_{\theta_t}(x_i)}$$

  Hence: $f_{\theta_t} = f_0 + \sum \vartheta_{i,t} \Theta^{(L)}(x, x_i)$.
- Final function: $f_0 +$ Kernel method for $\mathcal{C}_N(\cdot + f_0)$

MSE

- For MSE, $\frac{\partial \mathcal{C}_N}{\partial f_{\theta_t}(x_i)} = f_{\theta_t}(x_i) - y_i$: linear differential equation.
- On training points, the Gram matrix yields the speed of convergence.
- The function is Gaussian during training.
- Final function: $f_{\theta_\infty} = f_0 + \mathrm{KR}_{\lambda=0,(X,Y)}(f^* - f_0)$ or

$$f_{\theta_\infty} = \mathrm{KR}_{\lambda=0,(X,Y)}\left(f^*\right) + \epsilon,$$

  with **noise error term** $\epsilon = f_0 - \mathrm{KR}_{\lambda=0,(X,Y)}(f_0)$. [Zhang, Xu and al (2019)]

Answer to the Four Questions: train error and generalization?

- **Training:**

For MSE loss: training loss $= 0$. In general minimun error loss attained.

- **Generalization:**

Very large FCNN should generalize as RKHS methods: Rademacher bound should yield bounds of the form $\sqrt{\frac{Y\Theta^{-1}Y}{N}}$ for bounded Lip. cost. **[Arora, Du and al 2019 - 2 Layer ReLu and bounded Lip. cost function]**

Consequences: Training of large depth networks

Order-Chaos during inference [Daniely and al. 2016] [S.S. Schoenholz and al. 2017] [Hayou, Doucet, Rousseau 2019]
Depending on the variance of the initialisation as $L \to \infty$: $\Sigma^{(L)} \to C$ (order) or $\Sigma^{(L)} \to C_1 + C_2 \delta_{x=y}$ (chaos)

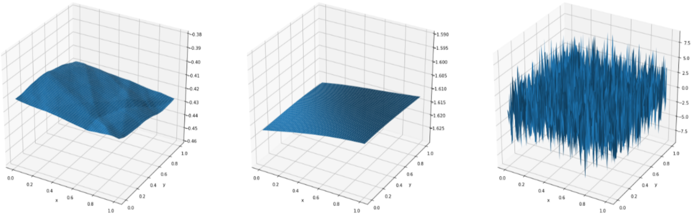


Figure: From “On the Impact of the Activation Function on Deep Neural Networks Training” [Hayou and al]

Consequences: Training of large depth networks

Freeze-Chaos during training [Jacot, Gabriel, Hongler 2019] [Agarwal, Awasthi, Kale 2020]
Depending on the variance of the initialisation:

- $\Theta^{(L)} \to C$ (order), the bias are two important, difficult to train.
- $\Theta^{(L)} \sim C_L \delta_{x=y}$ (chaos), easier to train, but generalization not good.

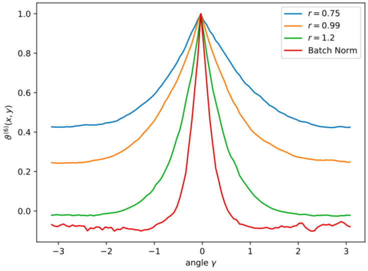


Figure: From "Order and Chaos: NTK views on DNN Normalization, Checkerboard and Boundary Artifacts" [A. Jacot, F. Gabriel, F. Ged, C. Hongler]

Consequences:Generalization

Function loss is convex : *noise in the predictor is bad.*

$$\mathbb{E}\left[\int (f_{\theta_\infty}(x) - f^*(x))^2 \, d\mu(x)\right] = \underbrace{\int (\mathbb{E}(f_{\theta_\infty}(x)) - f^*(x))^2 \, d\mu(x)}_{\text{Bias}} + \underbrace{\int \mathbb{V}\text{ar}\,[f_{\theta_\infty}(x)] \, d\mu(x)}_{\text{Variance}}$$

- The noise due to f_{θ_0} can be suppressed: train $f_\theta - f_{\theta_0}$ instead of f_θ
  - same dynamics + initialization $= 0 \to$ Kernel method
- ["Scaling description of generalization with number of parameters in deep learning", Geiger, Jacot, Spigler, **Gabriel**, Sagun, d'Ascoli, Biroli, Hongler, Wyart]
  Still noise due to fluctuations($\Theta^{(L)}_{t=0}$)$\sim P^{-\frac{1}{4}}$ and fluctuations($\Theta^{(L)}_t - \Theta^{(L)}_{t=0}$)$\sim P^{-\frac{1}{2}}$
  - Fluctuations of $f_{\theta_\infty}(x) \sim P^{-\frac{1}{4}}$, and Variance $\sim P^{-\frac{1}{2}}$,
  - If bias is constant in overparameterized regime:

$$\text{Generalization error} \sim \text{Error}_{P=\infty} + P^{-\frac{1}{2}}.$$

Double curve descent phenomenon

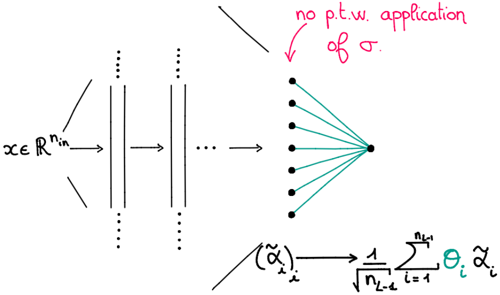
Extreme Learning and Regularized Kernel Method

Extreme Learning

Extreme Learning $=$ Learning the last layer's parameters.

Extreme Learning

Extreme Learning $=$ Learning the last layer's parameters.



▷ To simplify, we consider no bias (i.e. no additive parameter) for the last layer, and we assume that there is no pointwise application of the non-linearity at the last hidden layer.
▷ We assume that all hidden layers, except the last one, are infinite $\Longrightarrow f_i^{(L-1)}$ are i.i.d. $\mathcal{N}\left(0, \Sigma^{(L-1)}\right)$.
▷ We train only the last hidden layer, with a ℓ_2-norm penalization on $\boldsymbol{\theta}$.

Result : This is close to a Kernel Method with kernel $\Sigma^{(L-1)}$ but with a **larger regularisation.**

Implicit Regularization of Finite Sampling of Features

Rahimi & Recht's Random Features

$$f_{\boldsymbol{\theta}} : \mathbb{R}^{n_{in}} \xrightarrow{f} \mathbb{R}^{P} \xrightarrow[x \to \frac{1}{\sqrt{P}} \boldsymbol{\theta} x]{} \mathbb{R}$$

- f is an infinite neural network at initialization (recall: no pointwise application of σ for the output layer) **in particular,** $f = (f_j)_{j=1}^{P}$ **i.i.d. G.P.** $\mathcal{N}(0, K)$.
- The parameters are $\boldsymbol{\theta} \in \mathbb{R}^{P}$, and we consider N data points $(x_i, y_i)_{i=1}^{N}$.
- Optimization with $\frac{\lambda}{N} > 0$ penalization on the ℓ_2-norm of $\boldsymbol{\theta}$.

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \left(f_{\boldsymbol{\theta}}(x_i) - y_i \right)^2 + \frac{\lambda}{N} \|\boldsymbol{\theta}\|^2$$

- Closed Formulae: With $F_{ij} = \frac{1}{\sqrt{P}} f_j(x_i)$, optimal parameter: $\hat{\theta} = \left(F^T F + \lambda I_P \right)^{-1} F^T y$ leads to prediction: $\hat{y} = \underbrace{F \left(F^T F + \lambda I_P \right)^{-1} F^T}_{A_\lambda} y$ and optimal predictor:

$$\hat{f}_{\lambda}^{(RF)}(x) = \frac{1}{\sqrt{P}} \sum_{j=1}^{P} \hat{\theta}_j f_j(x).$$

Large number of features

$$\hat{y} = \underbrace{F\left(F^T F + \lambda I_P\right)^{-1} F^T}_{A_\lambda} y$$

But:

$$F\left(F^T F + \lambda I_P\right)^{-1} F^T = FF^T\left(FF^T + \lambda I_P\right)^{-1}$$

with

$$\left(FF^T\right)_{i,j} = \frac{1}{P}\sum_k f_k(x_i) f_k(x_j) \underset{P\to\infty}{\longrightarrow} K(x_i, x_j)$$

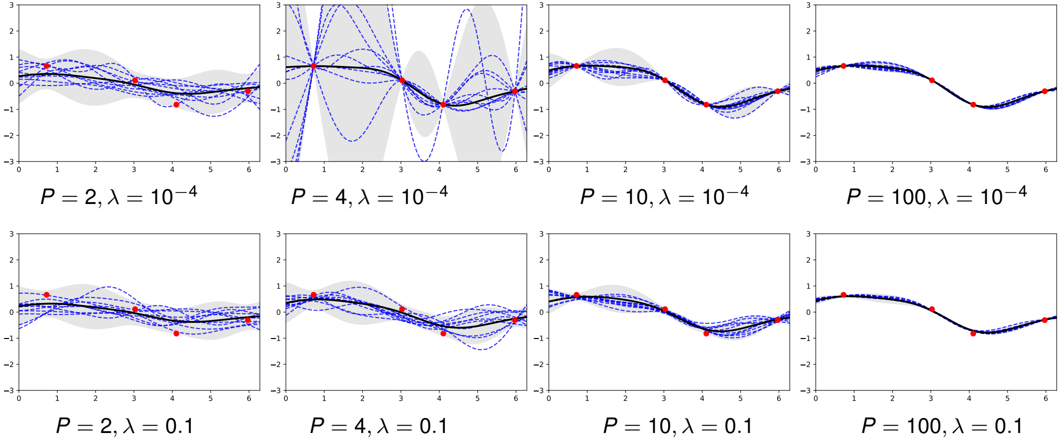
Thus:

$$\hat{y} \to K(X,X)\left[K(X,X) + \lambda I_N\right]^{-1} y$$

and the predictor converge to the K Kernel predictor with ridge λ:

$$\hat{f}_\lambda^{(RF)}(x) \to \hat{f}_\lambda^{(K)}(x) := K(x,X)\left[K(X,X) + \lambda I_N\right]^{-1} y.$$

R.F. Predictor



$P = 2, \lambda = 10^{-4}$ $P = 4, \lambda = 10^{-4}$ $P = 10, \lambda = 10^{-4}$ $P = 100, \lambda = 10^{-4}$

$P = 2, \lambda = 0.1$ $P = 4, \lambda = 0.1$ $P = 10, \lambda = 0.1$ $P = 100, \lambda = 0.1$

Finite number of features

$$\hat{y} = \underbrace{F\left(F^T F + \lambda I_P\right)^{-1} F^T}_{A_\lambda} y, \qquad \mathbb{E}\left[\hat{f}_\lambda^{(RF)}(x)\right] = \Sigma^{(L)}(x,X)\Sigma^{(L)}(X,X)^{-1}\mathbb{E}\left[A_\lambda\right] y$$

▷ The matrix A_λ can be studied using the **Stieljes transform**: $\frac{1}{P}\mathrm{Tr}\left[\left(F^T F + \lambda I_P\right)^{-1}\right]$

▷ The matrix F as a special structure: its columns are i.i.d. and Gaussian with cov $\frac{1}{P}K$:

$$F \sim \frac{1}{\sqrt{P}} K^{1/2} W^T$$

where W is a $P \times N$ random matrices with entries i.i.d. standard Gaussian.

▷ For the matrix $F^T F$:

$$F^T F \sim \frac{1}{P} W K W^T,$$

whose Stieljes transform can be studied like $K\left(\frac{1}{P} W^T W\right)$: product of a **Wishart Matrix** and a **deterministic matrix, well studied in free probability.**

Main result

Theorem (A. Jacot, B. Şimşek, F. Spadaro, C. Hongler, F. Gabriel, ICML 2020)

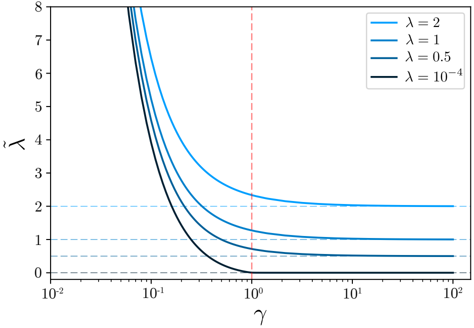
Even for $P < \infty$, $\mathbb{E}\left[\hat{f}_{\lambda}^{(RF)}(x)\right]$ is close to the Kernel predictor $\hat{f}_{\tilde{\lambda}}^{(K)}$ with a larger "effective ridge" $\tilde{\lambda}(\gamma, \lambda) > \lambda$ which is the unique solution of

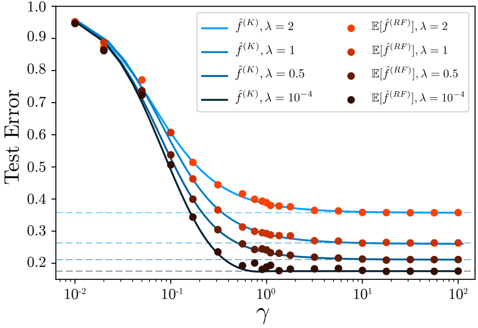
$$\tilde{\lambda} = \lambda + \frac{\tilde{\lambda}}{\gamma} \frac{1}{N} \mathrm{Tr}\left(K(X, X) \left(K(X, X) + \tilde{\lambda} \right)^{-1} \right),$$

where $K(X, X)$ is the Gram matrix of K.

*It is the **implicit regularization effect** of finite random features sampling.*

Effective Ridge and Test Error



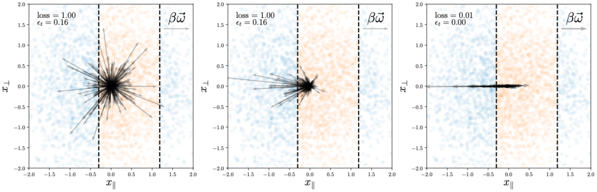


Kernel Method Generalization from the training set

Experiments on Structured Data and Finite N.N.

Recent pre-print of J. Paccolata, L. Petrinia, M. Geigera, K. Tylooa, and M. Wyart:
Setting: Classification with hinge Loss $c(y, y^*) = (1 - yy^*)^+$, shallow network, labels only depends on the first coordinate (stripe model), parameters initialized very small (feature learning regime).
Three phases during learning:

1. **Compressing Regime:** Parameters evolve independently and tend to align with the informative subspace.

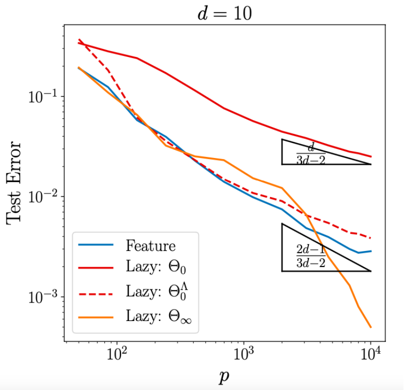


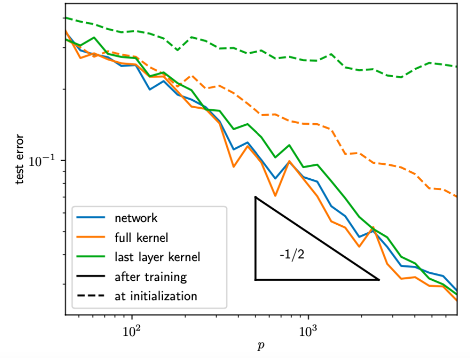
2. **Fitting regime:** when a fraction of constraints are satified, the N.N. tries to fit the labels but the parameters still evolve within the informative subspace.
3. **Over-fitting regime.**

Question: Is the N.T.K. theory no more interesting ?

Final NTK v.s. Neural Network

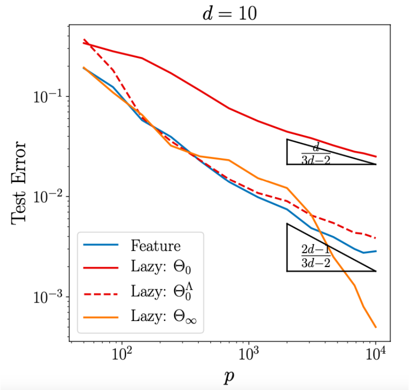
Answer: No, the kernel dynamics is still true but with an evolving kernel and in their experiment the NTK at the end of training is as good as the Neural Network !

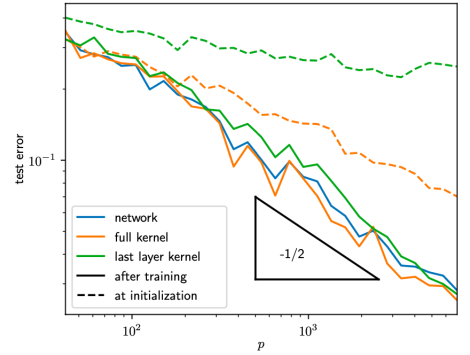




Final NTK v.s. Neural Network

Answer: No, the kernel dynamics is still true but with an evolving kernel and in their experiment the NTK at the end of training is as good as the Neural Network !

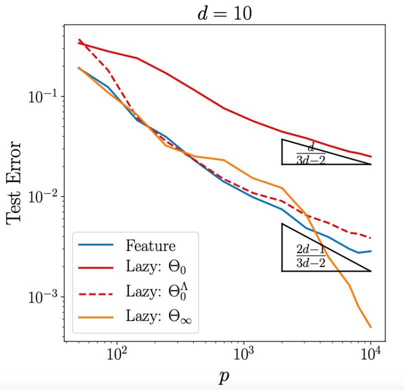


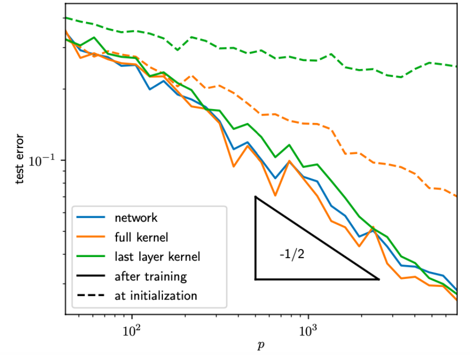


Question (Open): How does the N.T.K. improves ? How does the Neural Network "knows" that the kernel needs to evolve, based only on the training points ?

Final NTK v.s. Neural Network

Answer: No, the kernel dynamics is still true but with an evolving kernel and in their experiment the NTK at the end of training is as good as the Neural Network !





Question: Can we estimate the generalization error of a kernel method, based only on the training points ?

K.A.R.E. : the Kernel Alignement Risk Estimator

Consider a Kernel Method for

▹ Random i.i.d. training points $x_i \sim \mathcal{D}$ in a compact domain Ω

▹ Training labels $y_i^* = f^*(x_i) + \epsilon e_i$ with $e_i \sim \mathcal{N}(0,1)$

▹ Minimizing $\frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - y_i^*\right)^2 + \lambda \|f\|_{\mathcal{H}}^2$, i.e.

$$\hat{f}_\lambda(x) = K(x,X)\left[K(X,X) + \lambda I_N\right]^{-1} Y^*$$

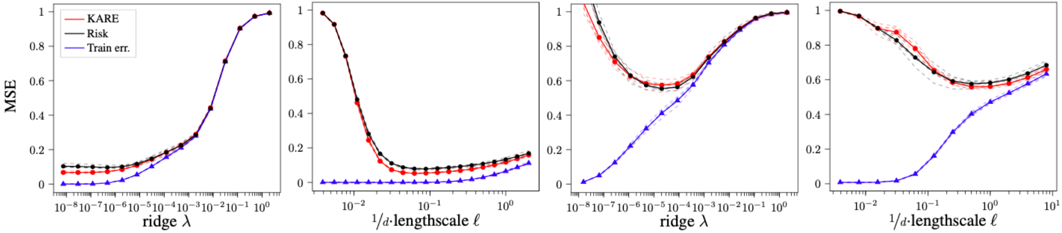
Fact (A. Jacot, B. Şimşek, F. Spadaro, C. Hongler, F. Gabriel, 2020)

We propose the following estimator of the Risk of the Kernel predictor $\hat{f}_\lambda$ with ridge λ :

$$\mathbb{E}_{x_1,\ldots,x_n,x\sim\mathcal{D}}\left[\left(\hat{f}_\lambda(x) - f^*(x)\right)^2\right] + \epsilon^2 \approx \frac{\frac{1}{N} Y^* \left[\frac{1}{N}K(X,X) + \lambda I_N\right]^{-2} Y^*}{\left(\frac{1}{N}\mathrm{Tr}\left[\left(\frac{1}{N}K(X,X) + \lambda I_N\right)^{-1}\right]\right)^2} = K.A.R.E.$$

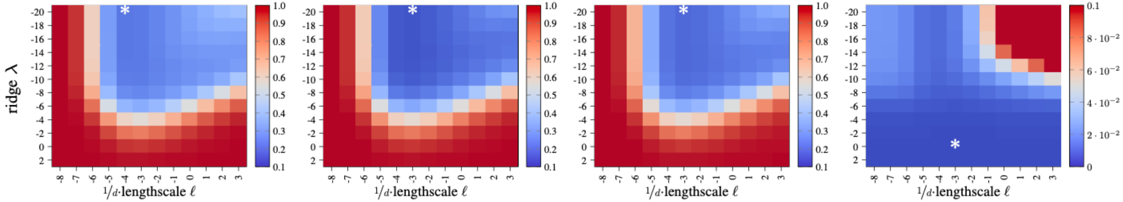
A theorem if only the second moments of the observations matters.

Experiments on Real Data



(a) MNIST, $\ell = d$ (b) MNIST, $\lambda = 10^{-5}$ (c) Higgs, $\ell = d$ (d) Higgs, $\lambda = 10^{-4}$

Hyperparameter selection with K.A.R.E.



(a) Risk (b) KARE Predictions (c) Cross Val. Predictions (d) Log-likelihood Estim.

Conclusion

To Wrap Up the Presentation

Recursive structure of A.N.N. $\Longrightarrow$ attractive properties. The training of A.N.N. using Gradient Descent with Random Initialization explained using the Neural Tangent Kernel. In the infinite width limit,

- the N.T.K. is **deterministic** and **constant during training,**
- the function follows a **Kernel Gradient Descent** with fixed kernel,
- the N.T.K. is > 0 and the **limiting dynamics converges to a global minimum,**
- the final function is of the form **Noise + Kernel Method**, and by a slight change on the definition of ANN, it becomes a **deterministic Kernel Method**.

In the finit width case:

- Fuctuations of the N.T.K. at initialization are the most important **and decrease with** P**: generalization error decreases as** $P^{-\frac{1}{2}}$ $\rightarrow$ **double curve descent is expected.**
- Last hidden layer finite followed by linear map, last parameters learned with ℓ^2 penalty with ridge λ: again **close to a Kernel Method with an other so-called "effective" ridge** $\tilde{\lambda} \geq \lambda$**.**

Kernel Method:

- Propose a **new estimator** for the **risk** for Kernel Methods: the **KARE**.

Thank you !

Bibliography

1. NTK: **Neural Tangent Kernel: Convergence and Generalization in Neural Networks**, NeuRIPS 2018, ***arXiv:1806.07572***, A. Jacot, F. Gabriel, C. Hongler

2. Generalization: **Scaling description of generalization with number of parameters in deep learning**, Journal of Statistical Mechanics: Theory and Experiment, ***arXiv:1901.01608***, M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d'Ascoli, G. Biroli, C. Hongler, M. Wyart

3. Order and Chaos: **Order and Chaos: NTK views on DNN Normalization, Checkeraboard and Boundary Artifacts**, ***arXiv:1907.05715***, A. Jacot, F. Gabriel, F. Ged, C. Hongler

4. Implicit Regularization: **Implicit Regularization of Random Feature Models**, ICML 2020, ***arXiv:2002.08404***, A. Jacot, B. Şimşek, F. Spadaro, C. Hongler, F. Gabriel

5. KARE: **Kernel Alignment Risk Estimator: Risk Prediction from Training Data**, ***arXiv:2006.09796***, A. Jacot, B. Şimşek, F. Spadaro, C. Hongler, F. Gabriel