Theoretic Fundamentals of Machine and Deep Learning

Neural Tangent Kernel

Aleksandr Petiushko

Lomonosov MSU, Faculty of Mechanics and Mathematics
MIPT, RAIRI
Nuro, Autonomy Interaction Research

Winter-Spring, 2023

AP

Content

1. Lazy regime
2. GD as PDE
3. Convergence rates
4. CNTK
5. NTK variations

AP

Introduction: other limiting approaches

Neural Network Gaussian Process (NNGP)

- Explicit connection to Bayesian Neural Nets (BNN) with layer width $\to \infty$
- Output of NN is approximated with Gaussian Process based on second moment of inputs
- No any training (and its dynamic) is included

Mean Field Theory (MFT)

- Higher order GD training dynamics is considered (not only linearization by the first order Taylor)
- No any explicit analytical formulations for the output if the number of layers is more than 2 (only existence theorems)
- No any practical solutions for number of layers more than 2

Introduction: NTK

Neural Tangent Kernel (NTK)

- GD training dynamics is considered
- Only linearization regime: weights during process are not changing much
- Has explicit analytical formulations for the output
- Has practical proof of concept even for CNN, but for small datasets / shallow networks

AP

NTK: the first approach

Main points of the original paper[1]:

- Behavior of DNN during GD is described by a related **Neural Tangent Kernel (NTK)**
- NTK only depends on the depth of the NN architecture, activation function and initialization variance
- Values of DNN outside the training set are described by NTK
- Behavior of wide DNN is close to the theoretical limit

[1] Jacot, Arthur, et al. "Neural tangent kernel: Convergence and generalization in neural networks." 2018

Lazy training

$$f(x,\theta) \approx f(x,\theta_0) + \langle \theta - \theta_0, \nabla_\theta f(x,\theta_0) \rangle$$

Key property (Lazy training):

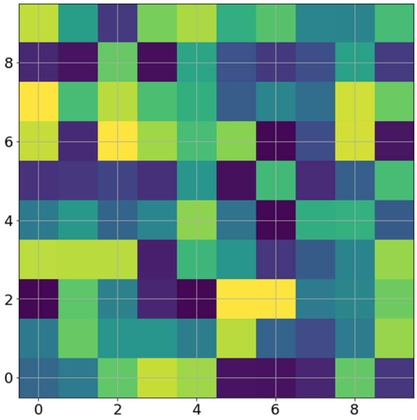
- Training loss is decreasing to 0 with minimal deviation of weights from their initialization (*linear dynamics*)
- In contrast to "**Mean-field**" regime, where weights evolve according to *non-linear dynamics*

AP

Lazy regime

Dynamics of NN weights during training:

- $W \in \mathbb{R}^{10\times10}$
- Gif source: `https://rajatvd.github.io/images/ntk/wim022_width10.gif`



AP

Definitions

- Training set: $(X, Y) = \{(x_i, y_i)_{i=1}^N\}, x \in \mathbb{R}^d, y \in \{1, \dots, c\}$
- Neural Net $f : \mathbb{R}^d \to \mathbb{R}^c$ (output: logits), parameterized by weights θ: f_θ
  - $f(x) = (z_1, \dots, z_c)^T$
- Loss function: $L : \mathbb{R}^c \to \mathbb{R}$
- Weights initialization: i.i.d. Gaussian $N(0, 1)$
- Signal propagation in the layer l: Wx is always multiplied by $\frac{1}{\sqrt{n_{l-1}}}$
  - This is not standard parameterization!
  - In the standard parameterization $\sigma^2 \sim \frac{1}{n_{l-1}}$, but Wx is used without any multiplication factor

AP

GD as a PDE

Consider iterative gradient descent (GD) formulation:

Iterative GD process

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L$$

Let's omit the learning rate η and reformulate it as a partial differential equation (PDE) $(\Delta t \to 0, \eta \to 0)$:

PDE

$$\frac{d\theta_t}{dt} = \dot{\theta}_t = -\nabla_{\theta_t} L = -\partial_z L \times \partial_{\theta_t} f$$

Remark. The last equality: the derivative of the composition of two functions by *chain rule*.

AP

Intuition: linear regression

- Regression task: $L = \frac{1}{2}\sum_{i=1}^{N}(f(x_i) - y_i)^2$
- Linear model: $f(x) = w^T x$
- Solution to regression task in matrix form: $w = (X^T X)^{-1} X^T Y$, $X \in \mathbb{R}^{N\times d}, Y \in \mathbb{R}^N$
  - Just by solving $\partial_w L = 0$

Suppose that we are solving this task by GD (as PDE):

- Dynamics of f:
  $\dot{f}_t = (\partial_{w_t} f_t)^T \times \dot{w}_t = (\partial_{w_t} f_t)^T \times (-\partial_{f_t} L \times \partial_{w_t} f_t) = -\sum_{i=1}^{N}(\partial_{w_t} f_t)^T \times (f_t - y) \times \partial_{w_t} f_t$
  - For linear regression in matrix form: $\dot{f}_t = -XX^T(f_t - Y)$

AP

Intuition: Regression and Constant Kernel

- Regression task: $L = \frac{1}{2}\sum_{i=1}^{N}(f(\theta_t, x_i) - y_i)^2$
- Empirical kernel: $\Theta_t(x_i, x_j) = \nabla_\theta f(\theta_t, x_i)^T \nabla_\theta f(\theta_t, x_j)$
- Dynamics of f: $\dot{f}_t(x) = -\Theta_t(x, X)(f_t(X) - Y)$ (*)
- Let us assume that the kernel is constant w.r.t. time: $\Theta_t(x_i, x_j) = \Theta_0(x_i, x_j)$
- Then $\dot{f}_t(X) = -\Theta_0(X, X)(f_t(X) - Y)$,
  - Solving the simple ODE $\dot{f} = c_1 f + c_2$ gives us $f_t(X) = f_0(X) - (I - e^{-\Theta_0(X,X)t})(f_0(X) - Y)$,
  - Substituting $f_t(X)$ in (*) leads to the dynamics solution $\dot{f}_t(x) = -\Theta_0(x, X)e^{-\Theta_0(X,X)t}(f_0(X) - Y)$
- And finally, solving another simple ODE $\dot{f} = c_3 e^{c_4 t}$, gives us $f_t(x) = f_0(x) - \Theta_0(x, X)\Theta_0^{-1}(X, X)(I - e^{-\Theta_0(X,X)t})(f_0(X) - Y)$,
- And if $f_0(X) = 0$, then $\boldsymbol{\lim_{t\to\infty} f_t(x) = \Theta_0(x, X)\Theta_0^{-1}(X, X)Y}$
- But what about non-constant kernel for different t?

AP

Reproducing Kernel Hilbert Space (RKHS)

RKHS[2]

$f, g \in RKHS$, $\|f - g\|$ is small $\Leftrightarrow$ $|f(x) - g(x)|$ is small for all x.

Representer Theorem[3]

- If a positive-definite real-valued kernel $k : X \times X \to \mathbb{R}$,
- If $f^* = \arg\min_{f \in RKHS\ H_k} \left[\frac{1}{N} \sum_{i=1}^{N} L(x_i, y_i, f(x_i))\right]$
- Then $f^*(\cdot) = \sum_{i=1}^{N} \alpha_i k(\cdot, x_i)$

[2] Reproducing Kernel Hilbert Space
[3] Representer Theorem

NTK: RKHS Solution

By **Representer Theorem** if $f^* = \arg\min_{f \in RKHS\ H_k} \left[\frac{1}{N}\sum_{i=1}^N L(x_i, y_i, f(x_i))\right] \Rightarrow$
$f^*(\cdot) = \sum_{i=1}^N \alpha_i k(\cdot, x_i)$

- Then for MSE loss in matrix form we have $L = \|Y - \Theta\alpha\|^2 \to \min_\alpha$, where $\Theta_{ij} = k(x_i, x_j) \Rightarrow \alpha = \Theta^{-1}Y$
- And the solution is $f^*(x) = k(x, X)\Theta^{-1}Y$

In our case $L(\theta_t) = \frac{1}{2}\sum_{i=1}^N (\langle \phi(x_i), \theta_t - \theta_0\rangle - y_i)^2$

- It means that we have the linearization of f with kernel $k(x, x') = \langle \phi(x), \phi(x')\rangle$, where $\phi(x) = \nabla_\theta f(\theta_0, x)$
- Note that for RKHS: $f(x) = \langle f(\cdot), k(\cdot, x)\rangle$ and $k(\cdot, x) = \phi(x)$
- $k(\cdot, \cdot)$ is positive-definite: $z^T \left[\nabla_\theta f^T \nabla_\theta f\right] z = (\nabla_\theta f z)^T (\nabla_\theta f z) = \|\nabla_\theta f z\|^2 \geq 0$

Result: $f^*(x) = k(x, X)\Theta^{-1}Y = \nabla_\theta f(\theta_0, x)^T \nabla_\theta f(\theta_0, X)\Theta^{-1}Y$

- where $\Theta = k(X, X) = \left[\nabla_\theta f(\theta_0, x_i)^T \nabla_\theta f(\theta_0, x_j)\right]_{i,j=1}^N$

AP

NTK: Results from original paper

- **Definition**: $\Theta_t : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^c \times \mathbb{R}^c$, $\Theta_t(x, x') = \mathbb{E}_{\theta_t}[\partial_{\theta_t} f(x) \partial_{\theta_t} f(x')]$
  - It is essentially the same as $\Theta_t(x, x') = \nabla_{\theta_t} f(x)^T \nabla_{\theta_t} f(x')$
  - **NB**: This is not a Hessian Matrix! Hessian $H = \left(\frac{\partial^2 f}{\partial_{\theta_i} \partial_{\theta_j}}\right)_{i,j=1}^n$, where $\theta \in \mathbb{R}^n$

Theorem 1

When the width of NN layers tends to infinity $n_l \to \infty$, then $\Theta_0 \to \Theta_\infty$, and this Θ_∞ depends only on:

- NN depth
- Non-linearity σ
- Variance at the initialization of θ

Theorem 2

$\forall t > 0$ when the width of NN layers tends to infinity $n_l \to \infty$, then $\Theta_t \to \Theta_0$

NTK: kernel for MLP

- Let the MLP has L layers, β is the scaling parameter for the bias
- Then we can compute by iterative procedure for $l = 1, \ldots, L-1$:
  $\Sigma^{(1)}(x, x') = \frac{1}{d} x^T x' + \beta^2$
  $\Sigma^{(l+1)}(x, x') = \mathbb{E}_{f \sim N(0, \Sigma^{(l)})}[\sigma(f(x))\sigma(f(x'))] + \beta^2$
  $\dot{\Sigma}^{(l+1)}(x, x') = \mathbb{E}_{f \sim N(0, \Sigma^{(l)})}[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))]$
  $\Theta_\infty^{(1)}(x, x') = \Sigma^{(1)}(x, x')$
  $\Theta_\infty^{(l+1)}(x, x') = \Theta_\infty^{(l)}(x, x')\dot{\Sigma}^{(l+1)}(x, x') + \Sigma^{(l+1)}(x, x')$
- And the final NTK is $\Theta_\infty = \Theta_\infty^{(L)}$

Remark1: Variation during training of individual activations in the hidden layers shrinks as their width grows.
Remark2: Overall variation of activations is significant, which allows the parameters of the lower layers to learn.

NTK: finite case[4]

- Let's move from limit theorems to more practical estimations based on finite layer width n_l and depth L ($m = \min_{1 \le l \le L} n_l$)
- Also let's use ReLU as the activation function: $\sigma(z) = \max(0, z)$

Theorem (Initialization)

Fix $\epsilon > 0$ and $\delta \in (0, 1)$. Suppose that $m \ge \Omega(\frac{L^{14}}{\epsilon^4} \log \frac{L}{\delta})$. Then for any inputs $x, x' \in \mathbb{R}^d$ such as $\|x\| \le 1, \|x'\| \le 1$, with probability at least $1 - \delta$ we have:

$$\left| \left\langle \partial_\theta f(\theta_0, x), \partial_\theta f(\theta_0, x') \right\rangle - \Theta_\infty(x, x') \right| \le \epsilon$$

Note: Error of approximation $\epsilon \sim m^{-\frac{1}{4}}$.

[4] Arora, Sanjeev, et al. "On exact computation with an infinitely wide neural net." 2019

NTK: finite case[5] (cont)

- Let's introduce some small positive multiplier $s > 0$ so as the initial output f is near 0: $f_{nn}(\theta, x) = sf(\theta, x)$
- Limit of output w.r.t. time: $f_{nn}(x) = \lim_{t\to\infty} f_{nn}(\theta_t, x)$
- NTK prediction: $f_{ntk}(x) = k(x, X)^T \Theta_\infty^{-1} Y$
- Denote $\lambda_0 = \lambda_{min}(\Theta_\infty)$

Theorem (Training)

Fix $\epsilon > 0$ and $\delta \in (0, 1)$ so as $\frac{1}{s} = poly(\frac{1}{\epsilon}, \log\frac{N}{\delta})$ and $m \geq poly(\frac{1}{s}, L, \frac{1}{\lambda_0}, N, \log\frac{1}{\delta})$. Then for any input $x \in \mathbb{R}^d$ such as $\|x\| \leq 1$, with probability at least $1 - \delta$ we have:

$$|f_{nn}(x) - f_{ntk}(x)| \leq \epsilon$$

Note: Error of approximation $\epsilon \sim poly(\frac{1}{m})$.

[5] Arora, Sanjeev, et al. "On exact computation with an infinitely wide neural net." 2019

CNTK[6]

- Let input image of size $P \times Q$
- $C^{(l)}$ — the number of channels on layer l,
- Convolutional filters are of size $q \times q$,
- c_σ — the inverse variance of $\sigma(x)$

Remark. The theory allows to have the **Global Average Pooling** (GAP) layer at the end, but not **MaxPooling** layers.

Time complexity is $O(N^2 P^2 Q^2 L)$.

CNTK formula. We let $\boldsymbol{x}, \boldsymbol{x}'$ be two input images.

- For $\alpha = 1, \ldots, C^{(0)}$, $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, define

$$\boldsymbol{K}_{(\alpha)}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}_{(\alpha)} \otimes \boldsymbol{x}'_{(\alpha)} \text{ and } \left[\boldsymbol{\Sigma}^{(0)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \sum_{\alpha=1}^{C^{(0)}} \operatorname{tr}\left(\left[\boldsymbol{K}_{(\alpha)}^{(0)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{\mathcal{D}_{ij,i'j'}}\right).$$

- For $h \in [L]$,
  - For $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, define

$$\boldsymbol{\Lambda}_{ij,i'j'}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}, \boldsymbol{x})\right]_{ij,ij} & \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} \\ \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}', \boldsymbol{x})\right]_{i'j',ij} & \left[\boldsymbol{\Sigma}^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}')\right]_{i'j',i'j'} \end{pmatrix} \in \mathbb{R}^{2\times 2}.$$

  - Define $\boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}'), \dot{\boldsymbol{K}}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{P\times Q\times P\times Q}$: for $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$,

$$\left[\boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \frac{c_\sigma}{q^2} \cdot \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}_{ij,i'j'}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right)} \left[\sigma(u)\, \sigma(v)\right],$$

$$\left[\dot{\boldsymbol{K}}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \frac{c_\sigma}{q^2} \cdot \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}_{ij,i'j'}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right)} \left[\dot{\sigma}(u)\, \dot{\sigma}(v)\right].$$

  - Define $\boldsymbol{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{P\times Q\times P\times Q}$: for $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$,

$$\left[\boldsymbol{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \operatorname{tr}\left(\left[\boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{D_{ij,i'j'}}\right).$$

1. First, we define $\boldsymbol{\Theta}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\Sigma}^{(0)}(\boldsymbol{x}, \boldsymbol{x}')$.
2. For $h = 1, \ldots, L-1$ and $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, we define

$$\left[\boldsymbol{\Theta}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{ij,i'j'} = \operatorname{tr}\left(\left[\dot{\boldsymbol{K}}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') \odot \boldsymbol{\Theta}^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{K}^{(h)}(\boldsymbol{x}, \boldsymbol{x}')\right]_{D_{ij,i'j'}}\right).$$

3. For $h = L$, we define $\boldsymbol{\Theta}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') = \dot{\boldsymbol{K}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \odot \boldsymbol{\Theta}^{(L-1)}(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{K}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')$.
4. The final CNTK value is defined as $\operatorname{tr}\left(\boldsymbol{\Theta}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right)$.

[6] Arora, Sanjeev, et al. "On exact computation with an infinitely wide neural net." 2019

CNTK: results

- Still 5-6% performance gap between the best CNTK and the best CNN
- For some depth values, CNTK provides better results than CNN

Depth	CNN-V	CNTK-V	CNTK-V-2K	CNN-GAP	CNTK-GAP	CNTK-GAP-2K
3	59.97%	64.47%	40.94%	63.81%	70.47%	49.71%
4	60.20%	65.52%	42.54%	80.93%	75.93%	51.06%
6	64.11%	66.03%	43.43%	83.75%	76.73%	51.73%
11	69.48%	65.90%	43.42%	82.92%	**77.43%**	51.92%
21	75.57%	64.09%	42.53%	83.30%	77.08%	52.22%

Table 1: Classification accuracies of CNNs and CNTKs on the CIFAR-10 dataset. CNN-V represents vanilla CNN and CNTK-V represents the kernel corresponding to CNN-V. CNN-GAP represents CNN with GAP and CNTK-GAP represents the kernel correspondong to CNN-GAP. CNTK-V-2K and CNTK-GAP-2K represent training CNTKs with only 2,000 training data.

AP

NTK: better convergence for the finite case[7]

- The linearization: $f_t^{lin}(x) = f_0(x) + \nabla_\theta f_\theta(x)|_{\theta=\theta_0}(\theta_t - \theta_0)$

Theorem

Let $n_1 = \cdots = n_L = m$ and assume $\lambda_{min}(\Theta) > 0$. Applying gradient descent with learning rate $\eta < \eta_{critical}$ (or gradient flow), for every $x \in \mathbb{R}^d$ with $\|x\|_2 \le 1$, with probability arbitrarily close to 1 over random initialization:

$$\sup_{t\ge0}\left\|f_t(x) - f_t^{lin}(x)\right\|_2, \sup_{t\ge0}\frac{\|\theta_t - \theta_0\|_2}{\sqrt{m}}, \sup_{t\ge0}\|\Theta_t - \Theta_0\|_F = O(m^{-\frac{1}{2}}) \quad m \to \infty$$

Note: Error of approximation $\epsilon \sim m^{-\frac{1}{2}}$.

[7] Lee, Jaehoon, et al. "Wide neural networks of any depth evolve as linear models under gradient descent." 2019

NTK: what about other loss functions[8]

- Consider cross entropy loss: $L(z, y) = -\sum_{i=1}^{c} y^i \log \sigma(z_i)$,
- Where SoftMax output $\sigma(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{c} \exp(z_j)}$ and $\partial_{z_i} L = \sigma(z_i) - y_i$
- Taking into account GD in the form of PDE, the dynamics is $\dot{f}_t^i(x) = \nabla_\theta f_t^i(x) \dot{\theta}_t = -\nabla_\theta f_t^i(x) \sum_{j=1} \sum_{(x',y)} \nabla_\theta f_t^j(x')^T \partial_{z_j} L(z, y) = -\sum_{(x',y)} \sum_{j=1} \nabla_\theta f_t^i(x) \nabla_\theta f_t^j(x')^T (\sigma(z_j) - y_j)$
- Let us denote $\Theta_t^{ij}(x, X) = \nabla_\theta f_t^i(x) \nabla_\theta f_t^j(x')^T$
- Then the final result is $\dot{f}_t(x) = -\Theta_t(x, X)(\sigma(f_t(X)) - Y)$
- This is ODE. Unfortunately, no closed form solution, only numerical solving...

[8] Lee, Jaehoon, et al. "Wide neural networks of any depth evolve as linear models under gradient descent." 2019

NTK: how to use MSE loss for classification

Let us first construct ground truth answer y. Two approaches:

One-hot encoding

$y = (0, \ldots, 0, 1, 0, \ldots, 0)$, where 1 stands on the place of the correct class.

Zero-centered One-hot encoding

$y = (-\frac{1}{c}, \ldots, -\frac{1}{c}, \frac{c-1}{c}, -\frac{1}{c}, \ldots, -\frac{1}{c})$, where $\frac{c-1}{c}$ stands on the place of the correct class.

And after that use MSE loss function: $L(z, y) = (z - y)^2$, where $z \in \mathbb{R}^c$ — NN output.

NTK: parameterization

Q: But why we needed this factor $\frac{1}{\sqrt{m}}$ for signal propagation $\frac{1}{\sqrt{m}}Wx$, while initialization for W is $N(0,1)$?

- Suppose output of some layer is $z \in \mathbb{R}^m$, where m – layer width
- NNGP kernel $K_z = \mathbb{E}_\theta[z^T z]$
- NTK kernel $\Theta_z = \mathbb{E}_\theta[\frac{\partial z^T}{\partial \theta}\frac{\partial z}{\partial \theta}]$
- Let $y = \frac{1}{\sqrt{m}}Wz$, where $W \in \mathbb{R}^{m\times m}, y \in \mathbb{R}^m$
- Then $K_y = \mathbb{E}_{W,\theta}[y^T y] = \frac{1}{m}\mathbb{E}_{W,\theta}[z^T W^T W z]$
- Using i.i.d. assumption $\frac{1}{m}\mathbb{E}_W[W^T W] = \frac{1}{m} \times m \times I_{m\times m} = I_{m\times m}$
- As the result, $K_y = \mathbb{E}_\theta[z^T z] = K_z$, and no dependency on $m \to \infty$!
- The same calculations for $\Theta_y = K_z + \Theta_z$

AP

NTK: parameterization comparison

- Let us qualitatively compare the parameterization/initialization schemes
- NTK regime: $\dot{\theta} \to 0, \Theta = const$ if $m \to \infty$
- Standard regime: $\dot{\theta} \neq 0, \Theta \to \infty$ if $m \to \infty$

Parameterization	Standard (naive)	NTK
Layer equation, $x^{l+1} =$	$W^l x^l + b^l$	$\frac{\sigma_w}{\sqrt{sN^l}} W^l x^l + \sigma_b b^l$
Weight shape, $W^l \in$	$\mathcal{R}^{sN^{l+1} \times sN^l}$	
W initialization, $W_{ij}^l \sim$	$\mathcal{N}\left(0, \frac{\sigma_w^2}{sN^l}\right)$	$\mathcal{N}(0,1)$
b initialization, $b_i^l \sim$	$\mathcal{N}\left(0, \sigma_b^2\right)$	$\mathcal{N}(0,1)$
NTK, $s \to \infty, \Theta^{l+1} =$	diverges	$\sigma_w^2 K^l + \sigma_b^2 + \sigma_w^2 \Theta^l$

AP

NTK: current challenges[9]

Architecture → Dataset size ↓	Fully-connected	CNNs	CNNs w/ pooling
O(100)	🥱	🤗	🙂
O(10,000)	CIFAR10: **O(0.1) GPU-hours**	CIFAR10: **O(1) GPU-hours**	CIFAR10: **O(1000) GPU-hours**
O(1,000,000)	🤔	😱	☠️

[9]Image source: `https://iclr.cc/virtual_2020/poster_SklD9yrFPS.html`

AP

Takeaway notes

- Lazy regime: very powerful tool
- Can (sometimes!) analytically solve the dynamics of linearized NN training
- Dependence on the training set size :(

Thank you!