Neural Tangent Kernel and Over-parameterization

Speaker: Wenhao Yang

Outline

- **Neural Tangent Kernel (NTK) Perspective**
- **Over-parameterization: Convergence and Generalization**

Neural Tangent Kernel

- Suppose $f(\theta, x)$ is the output of a neural network
- Loss: $\ell(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(f(\boldsymbol{\theta}, \boldsymbol{x}_i) - y_i\right)^2$
- Gradient descent: $\frac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t} = -\nabla\ell(\boldsymbol{\theta}(t)) = -\sum_{i=1}^{n}\left(f(\boldsymbol{\theta}(t), \boldsymbol{x}_i) - y_i\right)\frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}$
- What is the evolution of $\boldsymbol{u}(t) = \left(f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)\right)_{i\in[n]}$

$$\frac{\mathrm{d}f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\mathrm{d}t} = \left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t}\right\rangle = -\sum_{j=1}^{n}\left(f(\boldsymbol{\theta}(t), \boldsymbol{x}_j) - y_j\right)\left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_j)}{\partial \boldsymbol{\theta}}\right\rangle$$

Denote $[\boldsymbol{H}(t)]_{i,j} = \left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_j)}{\partial \boldsymbol{\theta}}\right\rangle$

Random feature map

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -\boldsymbol{H}(t)\cdot(\boldsymbol{u}(t) - \boldsymbol{y})$$

[On exact computation with an infinitely wide neural net]

Neural Tangent Kernel

- Consider a simple setting, what if $\boldsymbol{H}(t) = \boldsymbol{H}^*$ is a constant kernel?

Why we assume like this?

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -\boldsymbol{H}^* \cdot (\boldsymbol{u}(t) - \boldsymbol{y}).$$

$$\Downarrow$$

$$u(t) = y + e^{-tH^*}(u(0) - y)$$

$$\Downarrow$$

Do spectral decomposition on $\boldsymbol{H}^*$

Only the kernel (null-space) of H^* is remained as $t \to \infty$

$$f^*(\boldsymbol{x}) = (\ker(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \ker(\boldsymbol{x}, \boldsymbol{x}_n)) \cdot (\boldsymbol{H}^*)^{-1}\boldsymbol{y}.$$

Neural Tangent Kernel

- L-hidden-layer fully-connected neural network

$$\boldsymbol{f}^{(h)}(\boldsymbol{x}) = \boldsymbol{W}^{(h)}\boldsymbol{g}^{(h-1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h}, \quad \boldsymbol{g}^{(h)}(\boldsymbol{x}) = \sqrt{\frac{c_\sigma}{d_h}}\sigma\left(\boldsymbol{f}^{(h)}(\boldsymbol{x})\right) \in \mathbb{R}^{d_h}, \qquad h = 1, 2, \ldots, L,$$

$\boldsymbol{W}^{(h)} \in \mathbb{R}^{d_h \times d_{h-1}}$ is the weight matrix, σ is the activation function, $c_\sigma = \left(\mathbb{E}_{z\sim\mathcal{N}(0,1)}\left[\sigma(z)^2\right]\right)^{-1}$

- What's the behavior of $[\boldsymbol{H}(t)]_{i,j} = \left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_j)}{\partial \boldsymbol{\theta}} \right\rangle$?

Theorem 3.1 (Convergence to the NTK at initializatoin). *Fix $\epsilon > 0$ and $\delta \in (0,1)$. Suppose $\sigma(z) = \max(0, z)$ and $\min_{h\in[L]} d_h \geq \Omega(\frac{L^6}{\epsilon^4}\log(L/\delta))$. Then for any inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d_0}$ such that $\|\boldsymbol{x}\| \leq 1, \|\boldsymbol{x}'\| \leq 1$, with probability at least $1-\delta$ we have:*

$$\left| \left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle - \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \right| \leq (L+1)\epsilon.$$

$H(t) \approx H(0) \approx H^*$

Weights i.i.d $\sim \mathcal{N}(0,1)$

Lipschitz guarantees convergence at initialization
Lipschitz + twice differential + bounded 2nd derivative guarantees convergence at training process

[On exact computation with an infinitely wide neural net]

Neural Tangent Kernel

- $$\boldsymbol{f}^{(h)}(\boldsymbol{x}) = \boldsymbol{W}^{(h)} \boldsymbol{g}^{(h-1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h}, \quad \boldsymbol{g}^{(h)}(\boldsymbol{x}) = \sqrt{\frac{c_\sigma}{d_h}} \sigma\left(\boldsymbol{f}^{(h)}(\boldsymbol{x})\right) \in \mathbb{R}^{d_h}, \qquad h = 1, 2, \ldots, L,$$
- $$\frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{W}^{(h)}} = \mathbf{b}^{(h)}(\boldsymbol{x}) \cdot \left(\boldsymbol{g}^{(h-1)}(\boldsymbol{x})\right)^\top, \qquad h = 1, 2, \ldots, L+1,$$
- $$\begin{aligned} \left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{W}^{(h)}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{W}^{(h)}} \right\rangle &= \left\langle \mathbf{b}^{(h)}(\boldsymbol{x}) \cdot \left(\boldsymbol{g}^{(h-1)}(\boldsymbol{x})\right)^\top, \mathbf{b}^{(h)}(\boldsymbol{x}') \cdot \left(\boldsymbol{g}^{(h-1)}(\boldsymbol{x}')\right)^\top \right\rangle \\ &= \left\langle \boldsymbol{g}^{(h-1)}(\boldsymbol{x}), \boldsymbol{g}^{(h-1)}(\boldsymbol{x}') \right\rangle \cdot \left\langle \mathbf{b}^{(h)}(\boldsymbol{x}), \mathbf{b}^{(h)}(\boldsymbol{x}') \right\rangle. \end{aligned}$$
- $$\begin{aligned} \mathbb{E}\left[\left[\boldsymbol{f}^{(h+1)}(\boldsymbol{x})\right]_i \cdot \left[\boldsymbol{f}^{(h+1)}(\boldsymbol{x}')\right]_i \middle| \boldsymbol{f}^{(h)}\right] &= \left\langle \boldsymbol{g}^{(h)}(\boldsymbol{x}), \boldsymbol{g}^{(h)}(\boldsymbol{x}') \right\rangle \\ &= \frac{c_\sigma}{d_h} \sum_{j=1}^{d_h} \sigma\left(\left[\boldsymbol{f}^{(h)}(\boldsymbol{x})\right]_j\right) \sigma\left(\left[\boldsymbol{f}^{(h)}(\boldsymbol{x}')\right]_j\right), \end{aligned}$$

Tend to Centered Gaussian process(CLT)
[Lee et al., 2018]

converge

$$\Sigma^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v) \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\sigma(u)\, \sigma(v)\right].$$

[Deep Neural Networks as Gaussian Processes]
[On exact computation with an infinitely wide neural net]

Neural Tangent Kernel

Theorem 3.1 (Convergence to the NTK at initializatoin). *Fix* $\epsilon > 0$ *and* $\delta \in (0,1)$. *Suppose* $\sigma(z) = \max(0,z)$ *and* $\min_{h\in[L]} d_h \geq \Omega(\frac{L^6}{\epsilon^4}\log(L/\delta))$. *Then for any inputs* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d_0}$ *such that* $\|\boldsymbol{x}\| \leq 1, \|\boldsymbol{x}'\| \leq 1$, *with probability at least* $1-\delta$ *we have:*

$$\left| \left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle - \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \right| \leq (L+1)\epsilon.$$

$$\Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h=1}^{L+1} \left(\Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \prod_{h'=h}^{L+1} \dot{\Sigma}^{(h')}(\boldsymbol{x}, \boldsymbol{x}') \right)$$

Neural Tangent Kernel

$$\Sigma^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}',$$

$$\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2},$$

$$\Sigma^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\sigma(u)\,\sigma(v)\right].$$

$$\dot{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right].$$

[On exact computation with an infinitely wide neural net]

Neural Tangent Kernel

- **Connection with kernel regression**

Theorem 3.2 (Equivalence between trained net and kernel regression). *Suppose* $\sigma(z) = \max(0, z)$, $1/\kappa = \operatorname{poly}(1/\epsilon, \log(n/\delta))$ *and* $d_1 = d_2 = \cdots = d_L = m$ *with* $m \geq \operatorname{poly}(1/\kappa, L, 1/\lambda_0, n, \log(1/\delta))$. *Then for any* $\boldsymbol{x}_{te} \in \mathbb{R}^d$ *with* $\|\boldsymbol{x}_{te}\| = 1$, *with probability at least* $1 - \delta$ *over the random initialization, we have*

$$|f_{nn}(\boldsymbol{x}_{te}) - f_{ntk}(\boldsymbol{x}_{te})| \leq \epsilon.$$

- $f_{ntk}(\boldsymbol{x}_{te}) = (\ker_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X}))^\top (\boldsymbol{H}^*)^{-1} \boldsymbol{y}$
- $[\ker_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X})]_i = \Theta^{(L)}(\boldsymbol{x}_{te}, \boldsymbol{x}_i)$
- $f_{nn}(\boldsymbol{x}_{te}) = \lim_{t\to\infty} f_{nn}(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})$

[On exact computation with an infinitely wide neural net]

Performance

- $n := n_1 = \ldots = n_{L-1}, n_L = 1$
- **Convergence of the NTK**

Target function: $f^*(x) = x_1 x_2$

NTK w.r.t. the final layer (L=4)
$x = (\cos\gamma, \sin\gamma)$
$x_0 = (1{,}0)$
10 independent initializations
Train on random generated data $\mathcal{N}(0{,}1)$

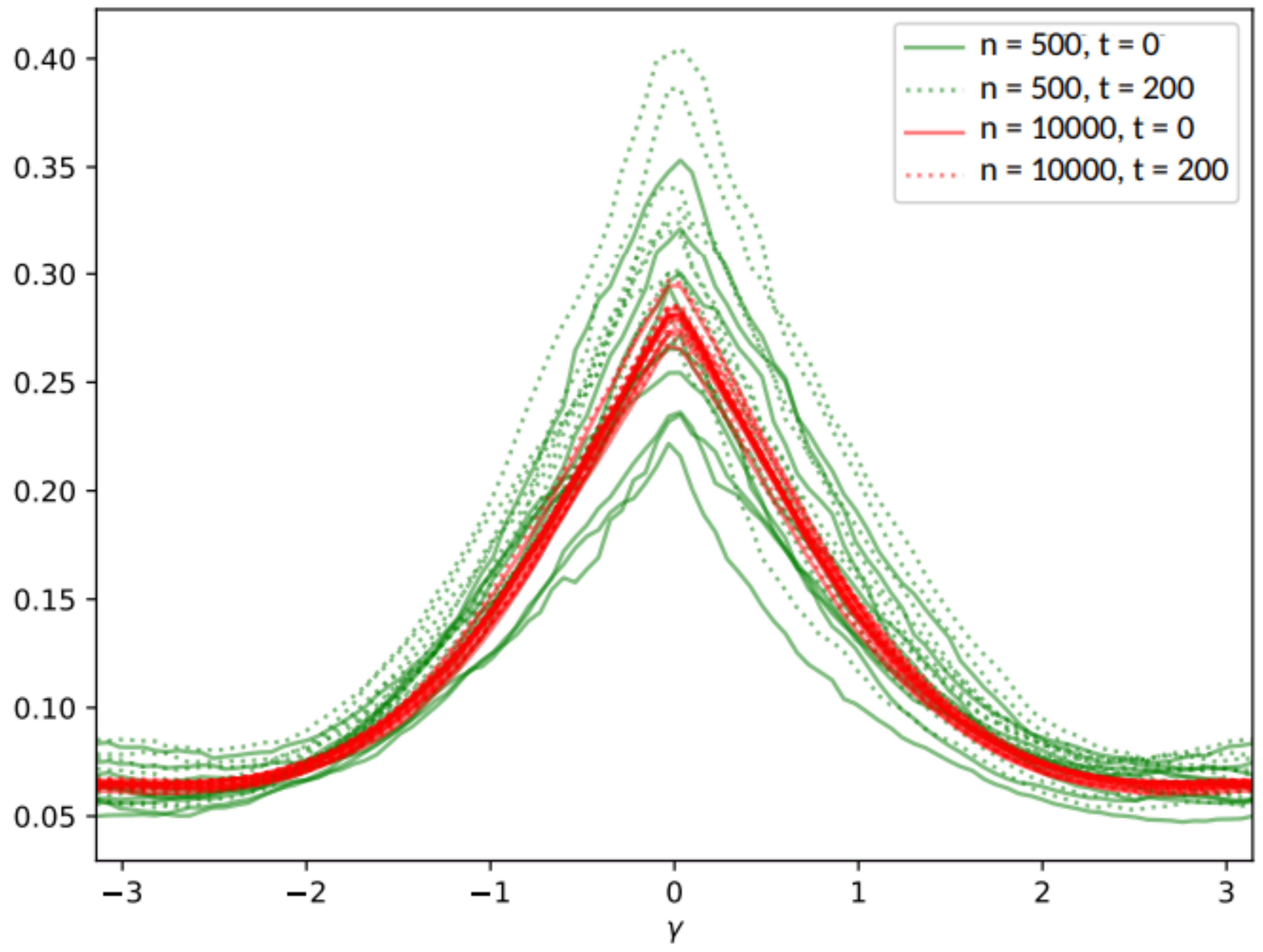


Figure 1: Convergence of the NTK to a fixed limit for two widths n and two times t.

[Deep Neural Networks as Gaussian Processes]

Performance

- **Kernel Regression**

10 random initializations
Trained on 4 points of the unit circle for 1000 steps with lr=1.0

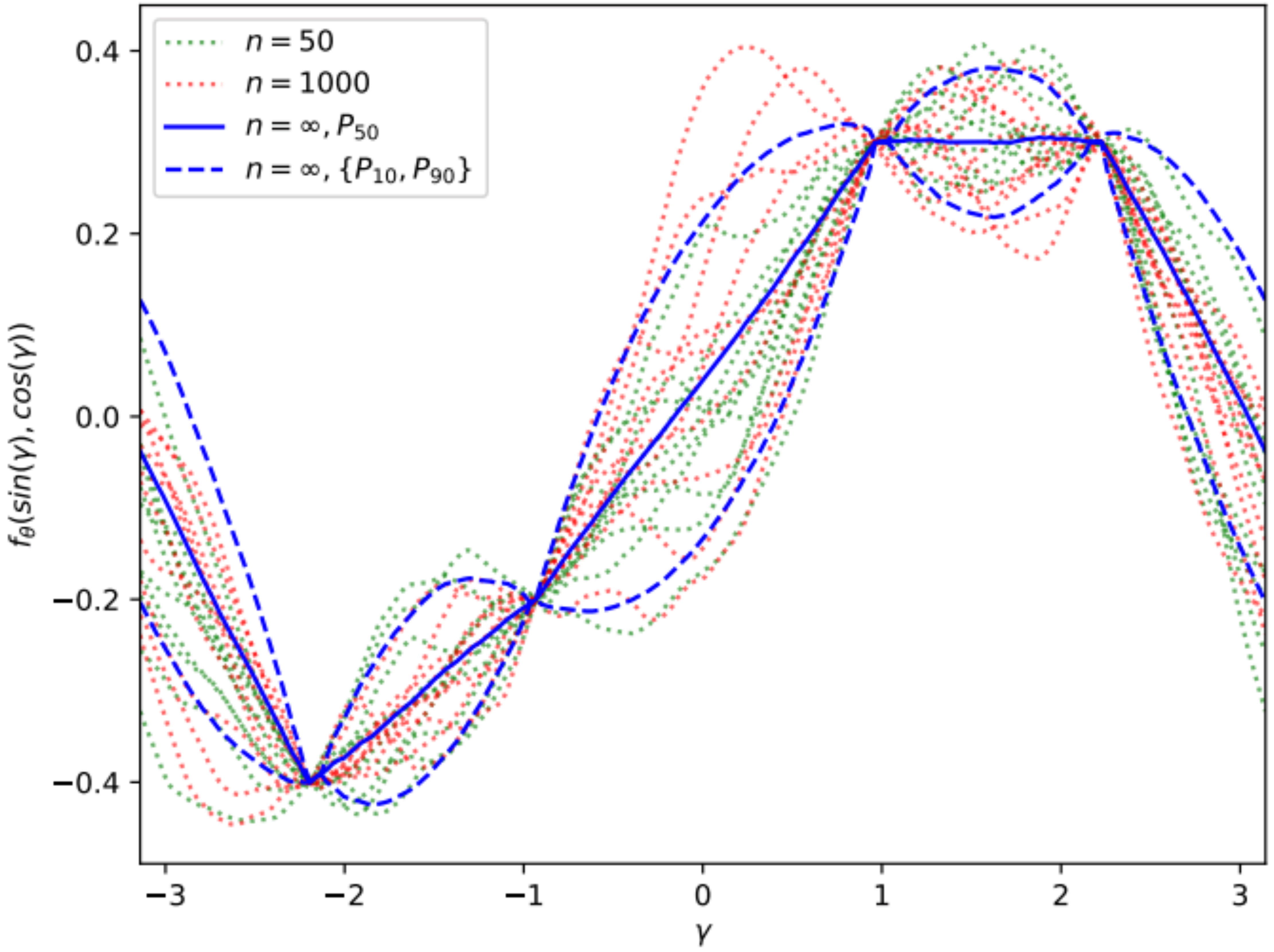


Figure 2: Networks function f_θ near convergence for two widths n and 10th, 50th and 90th percentiles of the asymptotic Gaussian distribution.

[Deep Neural Networks as Gaussian Processes]

NTK: Some properties

- **2-layer ReLU networks**

$$f(x;\theta) = \sqrt{\tfrac{2}{m}}\sum_{j=1}^{m} v_j\sigma(w_j^\top x), \text{ where } \sigma(u) = \max(0,u)$$

- **NTK:**

$$\begin{aligned} K(x,x') &= 2(x^\top x')\,\mathbb{E}_{w\sim\mathcal{N}(0,I)}[1\{w^\top x \geq 0\}1\{w^\top x' \geq 0\}] + 2\,\mathbb{E}_{w\sim\mathcal{N}(0,I)}[(w^\top x)_+(w^\top x')_+] \\ &= \|x\|\|x'\|\kappa\left(\frac{\langle x,x'\rangle}{\|x\|\|x'\|}\right) \end{aligned}$$

where $\kappa(u) := u\kappa_0(u) + \kappa_1(u)$

$$\kappa_0(u) = \frac{1}{\pi}(\pi - \arccos(u)), \qquad \kappa_1(u) = \frac{1}{\pi}\left(u\cdot(\pi - \arccos(u)) + \sqrt{1-u^2}\right)$$

- **Feature Maps**

$\varphi_{\mathcal{H},1} : \mathcal{H} \to \mathcal{H}_1$ w.r.t $(z,z') \in \mathcal{H}^2 \mapsto \|z\|\|z'\|\kappa_1(\langle z,z'\rangle/\|z\|\|z'\|)$ ➡ 1-Lipschitz

$\varphi_{\mathcal{H},0} : \mathcal{H} \to \mathcal{H}_0$ w.r.t $(z,z') \in \mathcal{H}^2 \mapsto \kappa_0(\langle z,z'\rangle/\|z\|\|z'\|)$

Lemma 1 (NTK feature map for fully-connected network). *The NTK for the fully-connected network can be defined as $K(x,x') = \langle\Phi_n(x),\Phi_n(x')\rangle$, with $\Phi_0(x) = \Psi_0(x) = x$ and for $k \geq 1$,*

$$\Psi_k(x) = \varphi_1(\Psi_{k-1}(x))$$

$$\Phi_k(x) = \begin{pmatrix}\varphi_0(\Psi_{k-1}(x)) \otimes \Phi_{k-1}(x) \\ \varphi_1(\Psi_{k-1}(x))\end{pmatrix},$$

[On the inductive bias of neural tangent kernels]

NTK: Some properties

- **Smoothness**

As we have $|f(x) - f(y)| \leq \|f\|_{\mathcal{H}} \|\Phi(x) - \Phi(y)\|_{\mathcal{H}}.$

Can we say Φ is smooth? If so, f is smooth. No

Proposition 3 (Non-Lipschitzness). *The kernel mapping $\Phi(\cdot)$ of the two-layer NTK is not Lipschitz:*

$$\sup_{x,y} \frac{\|\Phi(x) - \Phi(y)\|_{\mathcal{H}}}{\|x - y\|} \to +\infty.$$

This is true even when looking only at points x, y on the sphere. It follows that the RKHS $\mathcal{H}$ contains unit-norm functions with arbitrarily large Lipschitz constant.

Instability is due to $\varphi_{\mathcal{H},0}$, which comes from gradients w.r.t. first layer weights
Can we fix it?

Proposition 4 (Smoothness for ReLU NTK). *We have the following smoothness properties:* → A weaker smoothness property

1. *For x, y such that $\|x\| = \|y\| = 1$, the kernel mapping φ_0 satisfies $\|\varphi_0(x) - \varphi_0(y)\| \leq \sqrt{\|x - y\|}$.*
2. *For general non-zero x, y, we have $\|\varphi_0(x) - \varphi_0(y)\| \leq \sqrt{\frac{1}{\min(\|x\|,\|y\|)}\|x - y\|}$.*
3. *The kernel mapping Φ of the NTK then satisfies*

$$\|\Phi(x) - \Phi(y)\| \leq \sqrt{\min(\|x\|, \|y\|)\|x - y\|} + 2\|x - y\|.$$

[On the inductive bias of neural tangent kernels]

NTK: Some properties

- **Smoothness**

$$K_\sigma(x, x') = \langle x, x' \rangle \, \mathbb{E}_{w \sim \mathcal{N}(0,1)}[\sigma'(\langle w, x \rangle)\sigma'(\langle w, x' \rangle)] + \mathbb{E}_{w \sim \mathcal{N}(0,1)}[\sigma(\langle w, x \rangle)\sigma(\langle w, x' \rangle)].$$

Proposition 9 (Lipschitzness for smooth activations)**.** *Assume that σ is twice differentiable and that the quantities $\gamma_j := \mathbb{E}_{u \sim \mathcal{N}(0,1)}[(\sigma^{(j)}(u))^2]$ for $j = 0, 1, 2$ are bounded, with $\gamma_0 > 0$. Then, for x, y on the unit sphere, the kernel mapping Φ_σ of K_σ satisfies*

$$\|\Phi_\sigma(x) - \Phi_\sigma(y)\| \leq \sqrt{(\gamma_0 + \gamma_1) \max\left(1, \frac{2\gamma_1 + \gamma_2}{\gamma_0 + \gamma_1}\right)} \cdot \|x - y\|.$$

Example: $\sigma(u) = e^{u-2}$ **Lip=**$\sqrt{3}$

$\sigma(u) = \log(1+e^u)$ **Lip=1.75(approximate)**

[On the inductive bias of neural tangent kernels]

NTK: Some properties

- **Approximation properties**

What does these results say?

ReLU NTK has better approximation properties

Proposition 5 (Mercer decomposition of ReLU NTK). *For any $x, y \in \mathbb{S}^{p-1}$, we have the following decomposition of the NTK κ:*

$$\kappa(\langle x, y\rangle) = \sum_{k=0}^{\infty} \mu_k \sum_{j=1}^{N(p,k)} Y_{k,j}(x) Y_{k,j}(y), \tag{10}$$

where $Y_{k,j}, j = 1, \ldots, N(p,k)$ are spherical harmonic polynomials of degree k, and the non-negative eigenvalues μ_k satisfy $\mu_0, \mu_1 > 0$, $\mu_k = 0$ if $k = 2j+1$ with $j \geq 1$, and otherwise $\mu_k \sim C(p) k^{-p}$ as $k \to \infty$, with $C(p)$ a constant depending only on p. Then, the RKHS is described by:

$$\mathcal{H} = \left\{ f = \sum_{k \geq 0, \mu_k \neq 0} \sum_{j=1}^{N(p,k)} a_{k,j} Y_{k,j}(\cdot) \quad \text{s.t.} \quad \|f\|_{\mathcal{H}}^2 := \sum_{k \geq 0, \mu_k \neq 0} \sum_{j=1}^{N(p,k)} \frac{a_{k,j}^2}{\mu_k} < \infty \right\}. \tag{11}$$

Corollary 6 (Sufficient condition for $f \in \mathcal{H}$). *Let $f : \mathbb{S}^{p-1} \to \mathbb{R}$ be an even function such that all i-th order derivatives exist and are bounded by η for $0 \leq i \leq s$, with $s \geq p/2$. Then $f \in \mathcal{H}$ with $\|f\|_{\mathcal{H}} \leq C(p)\eta$, where $C(p)$ is a constant that only depends on p.*

Corollary 7 (Approximation of Lipschitz functions). *Let $f : \mathbb{S}^{p-1} \to \mathbb{R}$ be an even function such that $f(x) \leq \eta$ and $|f(x) - f(y)| \leq \eta \|x - y\|$, for all $x, y \in \mathbb{S}^{p-1}$. There is a function $g \in \mathcal{H}$ with $\|g\|_{\mathcal{H}} \leq \delta$, where δ is larger than a constant depending only on p, such that*

$$\sup_{x \in \mathbb{S}^{p-1}} |f(x) - g(x)| \leq C(p)\eta \left(\frac{\delta}{\eta}\right)^{-1/(p/2-1)} \log\left(\frac{\delta}{\eta}\right).$$

[On the inductive bias of neural tangent kernels]

Over-parameterization: Convergence

- **Setup**

loss function: $\mathcal{L}(\boldsymbol{\theta})=\frac{1}{n}\sum_{i=1}^{n}\ell\left(f(\boldsymbol{\theta},\mathbf{x}_i),y_i\right)=\frac{1}{2n}\sum_{i=1}^{n}\left(f(\boldsymbol{\theta},\mathbf{x}_i)-y_i\right)^2$

θ and W are equal

L-layer neural network (matrix form): $\mathbf{f}_{\mathbf{W}}(\mathbf{x})=\mathbf{V}\sigma(\mathbf{W}_L\sigma(\mathbf{W}_{L-1}\cdots\sigma(\mathbf{W}_1\mathbf{x})\cdots))$

- **Gradient Descent (GD)**

$$\mathbf{W}_l^{(t+1)}=\mathbf{W}_l^{(t)}-\eta\nabla_{\mathbf{W}_l}L(\mathbf{W}^{(t)})$$

- **Stochastic Gradient Descent (SGD)**

$$\mathbf{W}_l^{(t+1)}=\mathbf{W}_l^{(t)}-\frac{\eta}{B}\sum_{s\in\mathcal{B}^{(t)}}\nabla_{\mathbf{W}_l}\ell\big(\mathbf{f}_{\mathbf{W}^{(t)}}(\mathbf{x}_s),\mathbf{y}_s\big)\ \text{for all } l\in[L]$$

- **Natural Gradient Descent (NGD)**

$$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)-\eta\mathbf{F}(\boldsymbol{\theta}(k))^{-1}\frac{\partial\mathcal{L}(\boldsymbol{\theta}(k))}{\partial\boldsymbol{\theta}(k)}$$

F is the Fisher information matrix

Example: if the predictive distribution is in the exponential family, $F=\mathbb{E}_{x_i}J_i'H_lJ_i$, where J_i is the Jacobian matrix and H_l is the Hessian of the loss l

Over-parameterization: Convergence

- **Gradient Descent**

Assumption 3.1. For any $\mathbf{x}_i$, it holds that $\|\mathbf{x}_i\|_2 = 1$ and $(\mathbf{x}_i)_d = \mu$, where μ is an positive constant.

Assumption 3.2. For any two different training data points $\mathbf{x}_i$ and $\mathbf{x}_j$, there exists a positive constant $\phi > 0$ such that $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \geq \phi$.

Guarantee NTK (gram matrix) is positive definite

Theorem 3.3. Under Assumptions 3.1 and 3.2, and suppose the number of hidden nodes per layer satisfies

$$m = \Omega\big(kn^8 L^{12} \log^3(m)/\phi^4\big). \tag{3.1}$$

Then if set the step size $\eta = O\big(k/(L^2 m)\big)$, with probability at least $1 - O(n^{-1})$, gradient descent is able to find a point that achieves ϵ training loss within

$$T = O\big(n^2 L^2 \log(1/\epsilon)/\phi\big)$$

Proposition 3.6. Under Assumption 3.1, define the Gram matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ as follows

$$\mathbf{H}_{ij} = \mathbb{E}_{\mathbf{w}\sim\mathcal{N}(0,\mathbf{I})}[\mathbf{x}_i^\top \mathbf{x}_j \sigma'(\mathbf{w}^\top \mathbf{x}_i)\sigma'(\mathbf{w}^\top \mathbf{x}_j)],$$

then the assumption $\lambda_0 = \lambda_{\min}(\mathbf{H}) > 0$ is equivalent to Assumption 3.2. In addition, there exists a sufficiently small constant C such that $\lambda_0 \geq C\phi n^{-2}$.

[An improved analysis of training over-parameterized deep neural networks]

Over-parameterization: Convergence

- **Stochastic Gradient Descent**

Theorem 3.8. Under Assumptions 3.1 and 3.2, and suppose the number of hidden nodes per layer satisfies

$$m = \Omega\big(kn^{17}L^{12}\log^3(m)/(B^4\phi^8)\big). \tag{3.2}$$

Then if set the step size as $\eta = O\big(kB\phi/(n^3 m\log(m))\big)$, with probability at least $1 - O(n^{-1})$, SGD is able to achieve ϵ expected training loss within

$$T = O\big(n^5\log(m)\log^2(1/\epsilon)/(B\phi^2)\big)$$

[An improved analysis of training over-parameterized deep neural networks]

Over-parameterization: Convergence

- **Natural Gradient Descent (NGD)**

$$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)-\eta\mathbf{F}(\boldsymbol{\theta}(k))^{-1}\frac{\partial\mathcal{L}(\boldsymbol{\theta}(k))}{\partial\boldsymbol{\theta}(k)}$$

Square loss

$$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)-\eta\mathbf{J}^{\top}\left(\mathbf{J}\mathbf{J}^{\top}\right)^{-1}(\mathbf{u}-\mathbf{y}),$$

Condition 1 (Full row rank of Jacobian matrix). *The Jacobian matrix* $\mathbf{J}(0)$ *at the initialization has full row rank, or equivalently, the Gram matrix* $\mathbf{G}(0)=\mathbf{J}(0)\mathbf{J}(0)^{\top}$ *is positive definite.*

Condition 2 (Stable Jacobian). *There exists* $0\leq C<\frac{1}{2}$ *such that for all parameters* $\boldsymbol{\theta}$ *that satisfy* $\|\boldsymbol{\theta}-\boldsymbol{\theta}(0)\|_2\leq\frac{3\|\mathbf{y}-\mathbf{u}(0)\|_2}{\sqrt{\lambda_{\min}(\mathbf{G}(0))}}$, *we have*

$$\|\mathbf{J}(\theta)-\mathbf{J}(0)\|_2\leq\frac{C}{3}\sqrt{\lambda_{\min}(\mathbf{G}(0))}.$$

Like Lipschitz smoothness asp in Opt theory
Imply the network is close to a linearized network

Theorem 1 (Natural gradient descent). *Let Condition 1 and 2 hold. Suppose we optimize with NGD using a step size* $\eta\leq\frac{1-2C}{(1+C)^2}$. *Then for* $k=0,1,2,\ldots$ *we have*

$$\|\mathbf{u}(k)-\mathbf{y}\|_2^2\leq(1-\eta)^k\|\mathbf{u}(0)-\mathbf{y}\|_2^2. \tag{5}$$

[Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks]

Over-parameterization: Convergence

- **Natural Gradient Descent (NGD)**

$$f(\mathbf{w}, \mathbf{a}, \mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \phi(\mathbf{w}_r^\top \mathbf{x}),$$ where $\mathbf{w}_r \sim \mathcal{N}(\mathbf{0}, \nu^2 \mathbf{I})$ $a_r \sim \textbf{unif}\,[\{-1, +1\}]$

Assumption 1. *For all i, $\|\mathbf{x}_i\|_2 = 1$ and $|y_i| = \mathcal{O}(1)$. For any $i \neq j$, $\mathbf{x}_i \nparallel \mathbf{x}_j$.*

Denote $\lambda_0 = \lambda_{\textbf{min}}(G^\infty)$ $$\mathbf{G}_{ij}^\infty = \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \nu^2 \mathbf{I})} \left[\mathbf{x}_i^\top \mathbf{x}_j \mathbb{I} \left\{ \mathbf{w}^\top \mathbf{x}_i \geq 0, \mathbf{w}^\top \mathbf{x}_j \geq 0 \right\} \right] = \mathbf{x}_i^\top \mathbf{x}_j \frac{\pi - \arccos(\mathbf{x}_i^\top \mathbf{x}_j)}{2\pi}.$$

Theorem 3 (Natural Gradient Descent for overparameterized Networks). *Under Assumption 1, if we i.i.d initialize $\mathbf{w}_r \sim \mathcal{N}(\mathbf{0}, \nu^2 \mathbf{I})$, $a_r \sim \mathrm{unif}[\{-1, +1\}]$ for $r \in [m]$, we set the number of hidden nodes $m = \Omega\left(\frac{n^4}{\nu^2 \lambda_0^4 \delta^3}\right)$, and the step size $\eta = \mathcal{O}(1)$, then with probability at least $1 - \delta$ over the random initialization we have for $k = 0, 1, 2, \ldots$*

$$\|\mathbf{u}(k) - \mathbf{y}\|_2^2 \leq (1 - \eta)^k \|\mathbf{u}(0) - \mathbf{y}\|_2^2. \tag{11}$$

Is $1/\lambda_0$ small (or polynomial)?

Theorem 5. *Under this assumption on the training data, with probability $1 - n\exp(-n^\beta/4)$,*

x_i i.i.d from unit sphere $$\lambda_0 \triangleq \lambda_{\min}(\mathbf{G}^\infty) \geq n^\beta/2, \;\; \textit{where } \beta \in (0, 0.5) \tag{16}$$

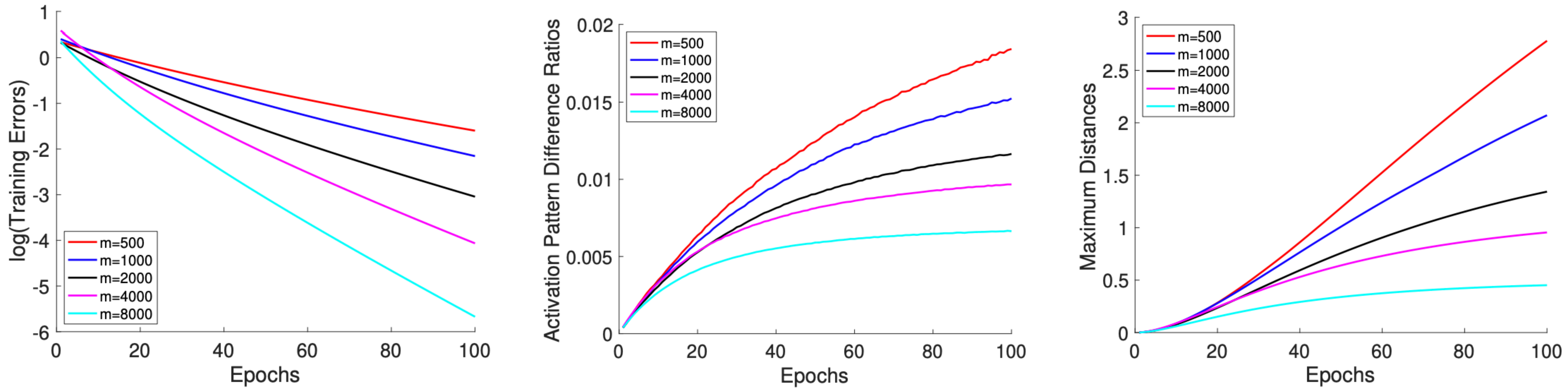
[Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks]

Over-parameterization: Convergence

- **2-layer NN, GD, opt first layer**

$$\frac{\sum_{i=1}^{m}\sum_{r=1}^{m}\mathbb{I}\left\{\text{sign}\left(\mathbf{w}_r(0)^\top\mathbf{x}_i\right)\neq\text{sign}\left(\mathbf{w}_r(k)^\top\mathbf{x}_i\right)\right\}}{mn}$$

$$\max_{r\in[m]}\left\|\mathbf{w}_r(k)-\mathbf{w}_r(0)\right\|_2$$



(a) Convergence rates.

(b) Percentiles of pattern changes.

(c) Maximum distances from initialization.

Figure 1: Results on synthetic data.

[Gradient Descent Provably Optimizes Over-Parameterized Neural Networks]

Over-parameterization: Generalization

- **Setup**

2-layer neural network: $$f_{\mathbf{W},\mathbf{a}}(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(\mathbf{w}_r^\top \mathbf{x}\right),$$

loss: $$\Phi(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - f_{\mathbf{W},\mathbf{a}}(\mathbf{x}_i)\right)^2$$

Gram matrix: $$\begin{aligned} \mathbf{H}_{ij}^{\infty} &= \mathbb{E}_{\mathbf{w}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\mathbf{x}_i^\top \mathbf{x}_j \mathbb{I}\left\{\mathbf{w}^\top \mathbf{x}_i \geq 0, \mathbf{w}^\top \mathbf{x}_j \geq 0\right\}\right] \\ &= \frac{\mathbf{x}_i^\top \mathbf{x}_j \left(\pi - \arccos(\mathbf{x}_i^\top \mathbf{x}_j)\right)}{2\pi}, \quad \forall i, j \in [n]. \end{aligned}$$

- **Results**

Theorem 3.2 (Informal version of Theorem 4.1). *With high probability we have:*

$$\Phi(\mathbf{W}(k)) \approx \frac{1}{2} \left\|(\mathbf{I} - \eta\mathbf{H}^{\infty})^k \mathbf{y}\right\|_2^2, \quad \forall k \geq 0.$$

Theorem 3.3 (Informal version of Theorem 5.1). *For any* 1*-Lipschitz loss function, the generalization error of the two-layer ReLU network found by GD is at most*

$$\sqrt{\frac{2\mathbf{y}^\top (\mathbf{H}^{\infty})^{-1}\mathbf{y}}{n}}. \tag{4}$$

[Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks]

Over-parameterization: Generalization

- **Setup**

$$f_{\mathbf{W}}(\mathbf{x}) = \sqrt{m} \cdot \mathbf{W}_L \sigma(\mathbf{W}_{L-1}\sigma(\mathbf{W}_{L-2} \cdots \sigma(\mathbf{W}_1 \mathbf{x}) \cdots)),$$

$$\min_{\mathbf{W}} L_{\mathcal{D}}(\mathbf{W}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} L_{(\mathbf{x},y)}(\mathbf{W}), \quad \text{where } L_{(\mathbf{x},y)}(\mathbf{W}) = \ell[y \cdot f_{\mathbf{W}}(\mathbf{x})] \qquad \ell(z) = \log[1+\exp(-z)]$$

Algorithm (SGD)

for $i = 1, 2, \ldots, n$ **do**
  Draw $(\mathbf{x}_i, y_i)$ from $\mathcal{D}$.
  Update $\mathbf{W}^{(i+1)} = \mathbf{W}^{(i)} - \eta \cdot \nabla_{\mathbf{W}} L_{(\mathbf{x}_i, y_i)}(\mathbf{W}^{(i)})$.
end for

- **Expected 0-1 Error Bound**

$$L_{\mathcal{D}}^{0-1}(\mathbf{W}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbb{1}\{y \cdot f_{\mathbf{W}}(\mathbf{x}) < 0\}]$$

Assumption 3.1. The data inputs are normalized: $\|\mathbf{x}\|_2 = 1$ for all $(\mathbf{x}, y) \in \mathrm{supp}(\mathcal{D})$.

Neural Tangent Random Feature $\mathcal{F}(\mathbf{W}^{(1)}, R) = \{f(\cdot) = f_{\mathbf{W}^{(1)}}(\cdot) + \langle \nabla_{\mathbf{W}} f_{\mathbf{W}^{(1)}}(\cdot), \mathbf{W}\rangle : \mathbf{W} \in \mathcal{B}(\mathbf{0}, R \cdot m^{-1/2})\},$

Bound: $m \geqslant \tilde{O}(\mathrm{poly}(R, L)) \cdot n^7 \cdot \log(1/\delta) \qquad \eta = \kappa \cdot R/(m\sqrt{n})$

$$\mathbb{E}[L_{\mathcal{D}}^{0-1}(\widehat{\mathbf{W}})] \leqslant \inf_{f \in \mathcal{F}(\mathbf{W}^{(1)}, R)} \left\{ \frac{4}{n} \sum_{i=1}^{n} \ell[y_i \cdot f(\mathbf{x}_i)] \right\} + \mathcal{O}\left[\frac{LR}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}} \right].$$

Trade off w.r.t R

If samples can be nearly fitted by NTRF, low generalization error is obtained

Small R, small $\mathscr{F}$, 1st term large, 2nd term small

If set $R = \tilde{\mathcal{O}}(1)$, 2nd term becomes $\tilde{\mathcal{O}}(n^{-1/2})$, only 1st term determines generalization

[Generalization bounds of stochastic gradient descent for wide and deep neural networks]

Over-parameterization: Generalization

- **NTRF and NTK**

$$m^{-1}\langle \nabla_{\mathbf{W}} f_{\mathbf{W}^{(1)}}(\mathbf{x}_i), \nabla_{\mathbf{W}} f_{\mathbf{W}^{(1)}}(\mathbf{x}_j)\rangle \xrightarrow{\mathbb{P}} \mathbf{\Theta}^{(L)}_{i,j},$$

Corollary 3.10. Let $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\lambda_0 = \lambda_{\min}(\mathbf{\Theta}^{(L)})$. For any $\delta \in (0, e^{-1}]$, there exists $\tilde{m}^*(\delta, L, n, \lambda_0)$ that only depends on δ, L, n and λ_0 such that if $m \geqslant \tilde{m}^*(\delta, L, n, \lambda_0)$, then with probability at least $1-\delta$ over the randomness of $\mathbf{W}^{(1)}$, the output of Algorithm 1 with step size $\eta = \kappa \cdot \inf_{\tilde{y}_i y_i \geqslant 1} \sqrt{\tilde{\mathbf{y}}^\top (\mathbf{\Theta}^{(L)})^{-1}\tilde{\mathbf{y}}}/(m\sqrt{n})$ for some small enough absolute constant κ satisfies

$$\mathbb{E}[L_{\mathcal{D}}^{0-1}(\widehat{\mathbf{W}})] \leqslant \tilde{\mathcal{O}}\left[L \cdot \inf_{\tilde{y}_i y_i \geqslant 1} \sqrt{\frac{\tilde{\mathbf{y}}^\top (\mathbf{\Theta}^{(L)})^{-1}\tilde{\mathbf{y}}}{n}}\right] + \mathcal{O}\left[\sqrt{\frac{\log(1/\delta)}{n}}\right],$$

Free with width, as long as width is large enough

where the expectation is taken over the uniform draw of $\widehat{\mathbf{W}}$ from $\{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}\}$.

Let $R = \tilde{\mathcal{O}}(\sqrt{\tilde{\mathbf{y}}^\top (\mathbf{\Theta}^{(L)})^{-1}\tilde{\mathbf{y}}})$

[Generalization bounds of stochastic gradient descent for wide and deep neural networks]

Over-parameterization: Generalization

- **Natural Gradient Descent (NGD)**

Theorem 6. *Given a target error parameter $\epsilon > 0$ and failure probability $\delta \in (0,1)$. Suppose $\nu = \mathcal{O}\left(\epsilon\sqrt{\lambda_0}\delta\right)$ and $m \geq \nu^{-2}\text{poly}\left(n, \lambda_0^{-1}, \delta^{-1}, \epsilon^{-1}\right)$. For any 1-Lipschitz loss function, with probability at least $1-\delta$ over random initialization and training samples, the two-layer neural network $f(\mathbf{w}, \mathbf{a})$ trained by NGD for $k \geq \Omega\left(\frac{1}{\eta}\log\frac{1}{\epsilon\delta}\right)$ iterations has expected loss $\mathcal{L}_{\mathcal{D}}(f(\mathbf{w}, \mathbf{a})) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell(f(\mathbf{w}, \mathbf{a}, \mathbf{x}), y)\right]$ bounded as:*

$$\mathcal{L}_{\mathcal{D}}(f(\mathbf{w}, \mathbf{a})) \leq \sqrt{\frac{2\mathbf{y}^\top(\mathbf{G}^\infty)^{-1}\mathbf{y}}{n}} + 3\sqrt{\frac{\log(6/\delta)}{2n}} + \epsilon \tag{18}$$

- **Where do these bounds come from?**

By Generalization bound with Rademacher Complexity, we only need to bound the complexity of function class

As width is large enough, we can approximate the complexity by kernel regression's result

$$\mathcal{F}_{A,B} = \{f(\mathbf{w}, \mathbf{a}) : \forall r \in [m], \|\mathbf{w}_r - \mathbf{w}_r(0)\|_2 \leq A, \|\mathbf{w} - \mathbf{w}(0)\|_2 \leq B\}$$

$$\mathcal{R}_S(\mathcal{F}_{A,B}) \leq \frac{B}{\sqrt{2n}}\left(1 + \left(\frac{2\log\frac{2}{\delta}}{m}\right)^{1/4}\right) + 2A^2\sqrt{m} + A\sqrt{2\log\frac{2}{\delta}}$$

[Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks]

Conclusion

- A **properly randomly initialized sufficiently wide** deep neural network **trained by gradient descent** with **infinitesimal** step size is **equivalent to a kernel regression predictor** with a **deterministic** kernel called neural tangent kernel (NTK).
- We can understand wide deep neural network by kernel regression. Can kernel regression (or classic machine learning theory) guide improvement and design on deep learning?
- Can extend to many other scenarios.
- Everything is just a kernel?
- Connection with Mean-field theory and ‘double descent’ phenomenon.
- Any benefits of non-linearity?
- Neural tangent models does not fully explain the success of neural network empirically. (mainly generalization error)

Table: Cifar10 experiments

Architecture	Classification error
Best convolutional NN	5%-
Best convolutional NT	23%
CNN of best CNT	19%

[https://stats385.github.io/ Song Mei’s slides]
[Ultra-Wide Deep Nets and Neural Tangent Kernel]

Reference

- NTK
  - Neural Tangent Kernel: Convergence and Generalization in Neural Networks
  - On exact computation with an infinitely wide neural net
  - On the inductive bias of neural tangent kernels
  - Deep Neural Networks as Gaussian Processes
  - Gradient Descent Provably Optimizes Over-Parameterized Neural Networks
- Convergence
  - An improved analysis of training over-parameterized deep neural networks
  - Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks
- Generalization
  - Generalization bounds of stochastic gradient descent for wide and deep neural networks
  - A generalization theory of gradient descent for learning over-parameterized deep relu network
  - Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Ne Networks
- General Introduction
  - https://rajatvd.github.io/NTK/
  - https://stats385.github.io/
  - Ultra-Wide Deep Nets and Neural Tangent Kernel