Scaling Neural Tangent Kernels via Sketching and Random Features

Insu Han

Yale University

Joint work with Amir Zandieh, Haim Avron,

Neta Shoham, Chaewon Kim, Jinwoo Shin

Outline

Introduction

- Kernel Regression
- Neural Tangent Kernel
- Existing Work

NTK Feature Map Construction

- Exact NTK Computation
- NTK Random Features
- NTK Sketch

Experiments

- Classification on MNIST datasets
- Classification on CIFAR-10 datasets

Conclusion

Outline

Introduction
- Kernel Regression
- Neural Tangent Kernel
- Existing Work

NTK Feature Map Construction
- Exact NTK Computation
- NTK Random Features
- NTK Sketch

Experiments
- Classification on MNIST datasets
- Classification on CIFAR-10 datasets

Conclusion

Kernel Regression

Kernel: a similarity function over pairs of data points in raw representation

- Mercer decomposition: for every kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$$

- ϕ is called a feature map

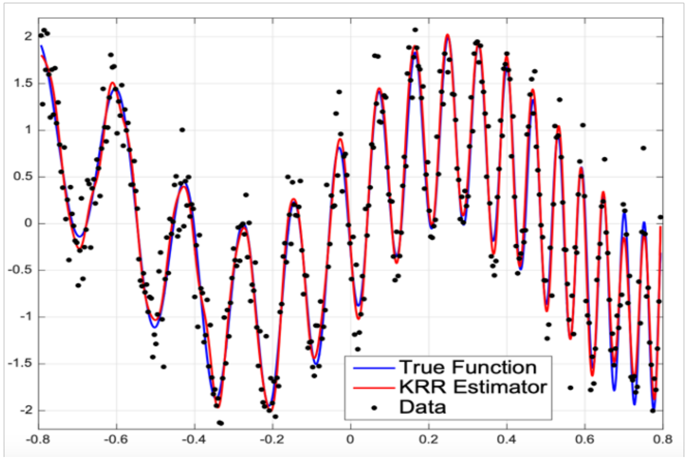
Kernel Regression

Kernel: a similarity function over pairs of data points in raw representation

– Mercer decomposition: for every kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$$

– ϕ is called a feature map



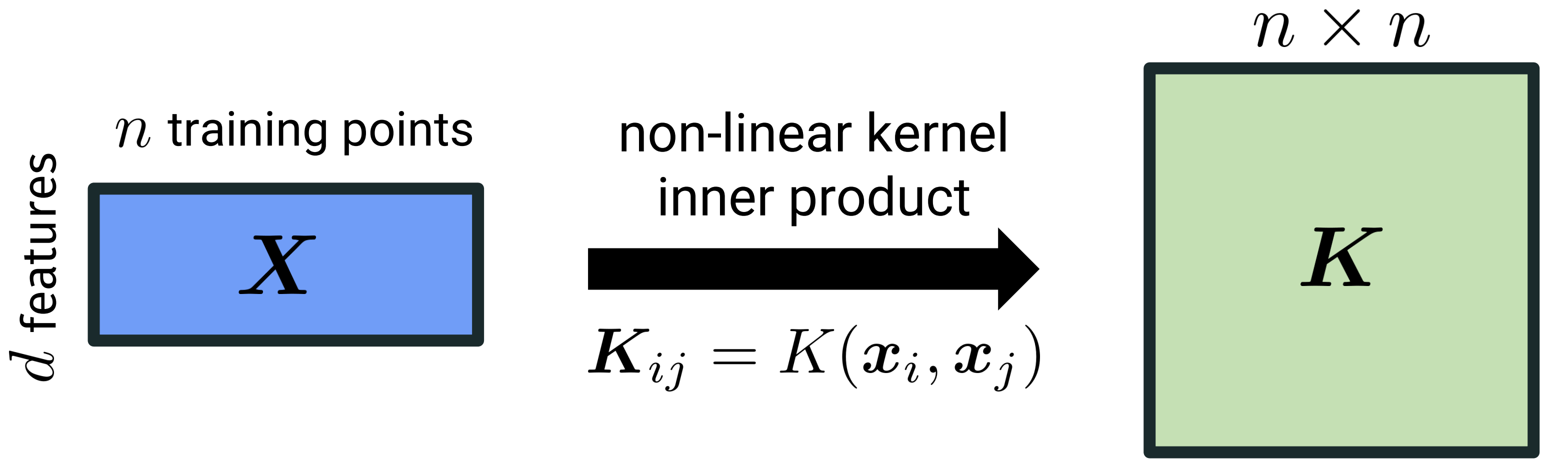
Kernel ridge regression:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \phi(\boldsymbol{x}_i)^\top \boldsymbol{w})^2 + \lambda \|\boldsymbol{w}\|_2^2$$

– Simple and yet powerful tool for learning non-linear relationships between data points

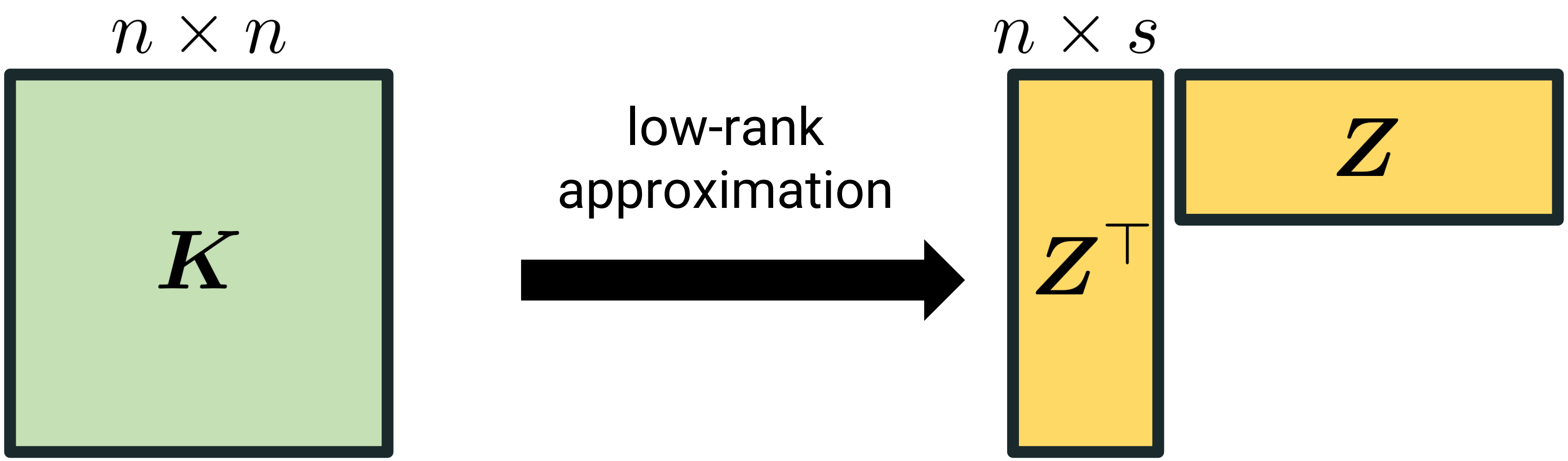
Scalability of Kernel Methods

Kernel methods are expensive



- Computing all kernel entries take $\Omega(n \cdot \mathrm{nnz}(\boldsymbol{X}) + n^2)$ time
- Even writing it down takes $\Omega(n^2 d)$ time and $\Omega(n^2)$ memory
- A single iteration of a linear system solver takes $\Omega(n^2)$ time
- For $n = 100{,}000$, $\boldsymbol{K}$ has 10 billion entries. Take 80 GB of storage!

Classical Solution: **Dimensionality Reduction**



Storing $\boldsymbol{Z}$ uses $\mathcal{O}(ns)$ space and computing $\boldsymbol{Z}^\top \boldsymbol{Z}\boldsymbol{x}$ takes $\mathcal{O}(ns)$ time

Orthogonalization, eigen-decomposition and pseudo-inversion of $\boldsymbol{Z}^\top \boldsymbol{Z}$ all take just $\mathcal{O}(ns^2)$ time

Neural Tangent Kernel (NTK)

Kernel: a similarity function over pairs of data points in raw representation

– Mercer kernel is a function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that for $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$$

– ϕ is called a feature map (can be a random function)

Tangent kernel:

– For a smooth function $f(\cdot, \theta)$, tangent kernel is an inner-product of gradient at a fixed θ_0:

$$K_T(\boldsymbol{x}, \boldsymbol{x}'; \theta_0) = \left\langle \left.\frac{\partial f(\boldsymbol{x}, \theta)}{\partial \theta}\right|_{\theta=\theta_0}, \left.\frac{\partial f(\boldsymbol{x}', \theta)}{\partial \theta}\right|_{\theta=\theta_0} \right\rangle$$

Neural Tangent Kernel

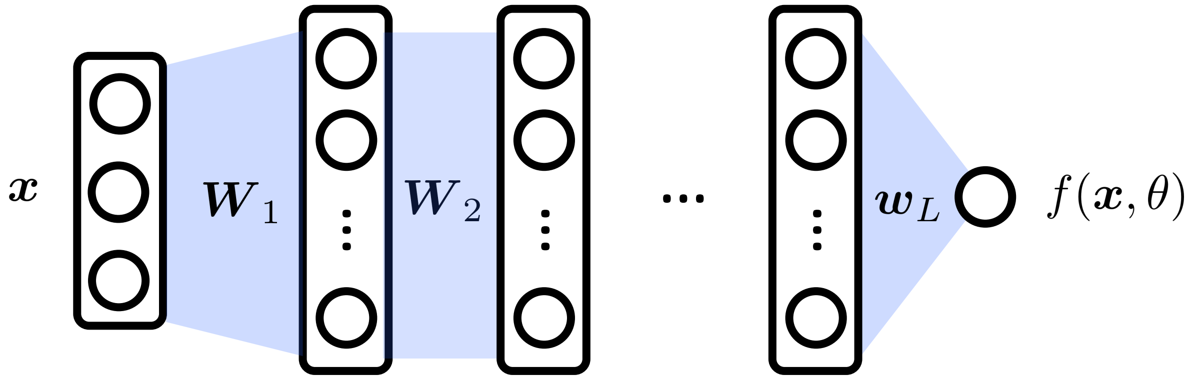
Fully-connected neural network:

$$f(\boldsymbol{x}, \theta) = \boldsymbol{h}_L^\top \boldsymbol{w}_L, \quad \boldsymbol{h}_\ell = \frac{c_\sigma}{\sqrt{d_\ell}} \sigma(\boldsymbol{h}_{\ell-1}^\top \boldsymbol{W}_\ell), \quad \boldsymbol{h}_0 = \boldsymbol{x}$$

- $\theta = (\boldsymbol{W}_1, \dots, \boldsymbol{w}_L)$: trainable parameters
- $c_\sigma = 1/\mathbb{E}_{z\sim\mathcal{N}}[\sigma(z)^2]$: normalization for activation

Neural Tangent Kernel [JGH18][LXS+19]**:**

$$K_{\mathrm{NTK}}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\theta\sim\mathcal{N}} \left\langle \frac{\partial f(\boldsymbol{x}, \theta)}{\partial \theta}, \frac{\partial f(\boldsymbol{x}', \theta)}{\partial \theta} \right\rangle$$



Neural Tangent Kernel

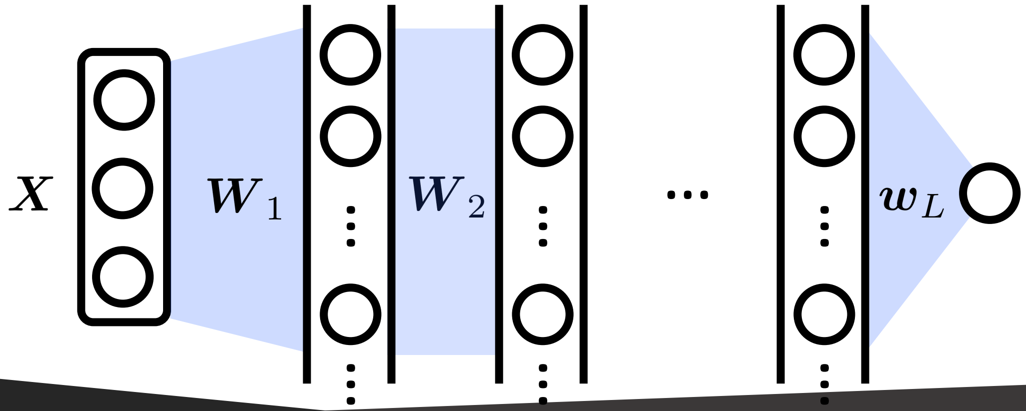
Under infinite width regime, neural network is equivalent to NTK

– Prediction of neural networks trained under ℓ_2-loss by gradient flow
– NTK solved by kernel regression

$$y_{\text{test}} = K_{\text{NTK}}(\boldsymbol{x}_{\text{test}}, \boldsymbol{X}) K_{\text{NTK}}(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y}$$

for training data $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top, \boldsymbol{y} \in \mathbb{R}^n$

– No need to learn parameter θ if $K_{\text{NTK}}(\cdot)$ is given



$$f(\boldsymbol{X}, \theta) = [f(\boldsymbol{x}_1, \theta), \ldots, f(\boldsymbol{x}_n, \theta)]^\top$$

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top$$

Neural Tangent Kernel

NTK is useful both in theory and practice:

- Generalization [CG19], optimization [ALS19], regularization [HLY20]
- In practice, NTK can perform better regression results than training neural networks (NN) [ADL+19]

Classifier	Friedman Rank	Average Accuracy	P90	P95	PMA
NTK	**28.34**	**81.95%±14.10%**	**88.89%**	**72.22%**	**95.72% ±5.17%**
NN (He init)	40.97	80.88%±14.96%	81.11%	65.56%	94.34% ±7.22%
NN (NTK init)	38.06	81.02%±14.47%	85.56%	60.00%	94.55% ±5.89%
RF	33.51	81.56% ±13.90%	85.56%	67.78%	95.25% ±5.30%
Gaussian Kernel	35.76	81.03% ± 15.09%	85.56%	**72.22%**	94.56% ±8.22%
Polynomial Kernel	38.44	78.21% ± 20.30%	80.00%	62.22%	91.29% ±18.05%

Comparisons of different classifiers on 90 UCI datasets [ADL+19]

Neural Tangent Kernel

NTK is useful both in theory and practice:

- Generalization [CG19], optimization [ALS19], regularization [HLY20]
- In practice, NTK can perform better regression results than training neural networks (NN) [ADL+19]

Computational issues:

- Even NTK can be computed exactly [ADH+19], solving kernel regression requires $\mathcal{O}(n^3)$ time $\Rightarrow$ infeasible for large n
- Convolution NTK (CNTK) with $h \times w$ images takes $\Omega(n^2 h^2 w^2)$ time

  E.g., Convolutional NTK with CIFAR-10 $\geq$ **1000** GPU hours [SFG+20]

Neural Tangent Kernel

NTK is useful both in theory and practice:

- Generalization [CG19], optimization [ALS19], regularization [HLY20]
- In practice, NTK can perform better regression results than training neural networks (NN) [ADL+19]

Computational issues:

- Even NTK can be computed exactly [ADH+19], solving kernel regression requires $\mathcal{O}(n^3)$ time $\Rightarrow$ infeasible for large n

Kernel approximation by feature map:

$$K_{\mathrm{NTK}}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle, \qquad \phi : \mathbb{R}^d \to \mathbb{R}^m$$

- Kernel ridge regression can be solved in $\mathcal{O}(nm^2 + m^3)$ time
- Memory complexity can be $\mathcal{O}(nm + m^2)$
- Usually, $m \ll n$

Existing Work

Goal: feature map construction ϕ such that $K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$

Gradient features of finitely wide network:

$$K_{\mathrm{NTK}}(\boldsymbol{x}, \boldsymbol{x}') := \mathop{\mathbb{E}}_{\theta' \sim \mathcal{N}} \left\langle \frac{\partial f(\boldsymbol{x}, \theta')}{\partial \theta}, \frac{\partial f(\boldsymbol{x}', \theta')}{\partial \theta} \right\rangle \qquad \text{(Definition)}$$

$$\approx \frac{1}{s} \sum_{i=1}^{s} \left\langle \frac{\partial f(\boldsymbol{x}, \theta_i)}{\partial \theta}, \frac{\partial f(\boldsymbol{x}', \theta_i)}{\partial \theta} \right\rangle$$

– $\theta_1, \dots, \theta_s$: random samples from Gaussian distribution
– Gradient features show poor performances in practice [ADH+19]

Contributions

Fast and accurate NTK approximation using Random Features/Sketching

	Feature dimension	Runtime
Gradient Features [ADH+19]	$\widetilde{\Omega}\left(\frac{L^{13}}{\varepsilon^8}\right)$	$\widetilde{\Omega}\left(\frac{L^{13}}{\varepsilon^8}\right)$
NTK Random Features (Our)	$\widetilde{\mathcal{O}}\left(\frac{L^{6}}{\varepsilon^4}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{13}}{\varepsilon^8}\right)$
NTK Sketch (Our)	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{11}}{\varepsilon^{6.7}}+\frac{L^3}{\varepsilon^2}d\right)$
CNTK Sketch (Our)	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{11}}{\varepsilon^{6.7}}\cdot(d_1 d_2)\right)$

*CNTK with $d_1 \times d_2$ images

– For all methods, the runtime of kernel regression reduces from $\mathcal{O}(n^3+n^2 d)$ to $\mathcal{O}(n(\mathtt{dimension}^2+\mathtt{runtime}))$

– For n images, the exact CNTK can be computed in $\mathcal{O}\left(n^2 L d_1^2 d_2^2\right)$ time while CNTK Sketch runs in $\widetilde{\mathcal{O}}\left(nL^{11} d_1 d_2 / \varepsilon^{6.7}\right)$ time

Contributions

Fast and accurate NTK approximation using Random Features/Sketching

	Feature dimension	Runtime
Gradient Features [ADH+19]	$\widetilde{\Omega}\left(\frac{L^{13}}{\varepsilon^8}\right)$	$\widetilde{\Omega}\left(\frac{L^{13}}{\varepsilon^8}\right)$
NTK Random Features (Our)	$\widetilde{\mathcal{O}}\left(\frac{L^{6}}{\varepsilon^4}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{13}}{\varepsilon^8}\right)$
NTK Sketch (Our)	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{11}}{\varepsilon^{6.7}} + \frac{L^3}{\varepsilon^2}d\right)$
CNTK Sketch (Our)	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{11}}{\varepsilon^{6.7}} \cdot (d_1 d_2)\right)$

*CNTK with $d_1 \times d_2$ images

Better performance than other heuristics under real-world datasets

- NTK Sketch with $L = 3$ achieves 72% of test accuracy of CIFAR-10 while the exact NTK, CNN achieve 70%, 64%, respectively
- CNTK Sketch is 190 times faster than the exact CNTK

Outline

Introduction
- Kernel Regression
- Neural Tangent Kernel
- Existing Work

NTK Feature Map Construction
- Exact NTK Computation
- NTK Random Features
- NTK Sketch

Experiments
- Classification on MNIST datasets
- Classification on CIFAR-10 datasets

Conclusion

Exact NTK Computation

NTK with ReLU activation [ADH+19]**:** for layer $\ell = 1, \ldots, L$ and $\boldsymbol{x} \in \mathbb{R}^d$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \underbrace{\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')}_{\text{function of } \Sigma^{(\ell-1)}}$$

where $K_{\mathrm{NTK}}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$

Exact NTK Computation

NTK with ReLU activation [ADH+19]**:** for layer $\ell = 1, \ldots, L$ and $\boldsymbol{x} \in \mathbb{R}^d$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

function of $\Sigma^{(\ell-1)}$

where $K_{\mathrm{NTK}}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$

- $\dot{\Sigma}^{(\ell)}, \Sigma^{(\ell)}$ can be expressed as a closed-form formula using $\Sigma^{(\ell-1)}$

$$\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = 1 - \frac{1}{\pi} \cos^{-1} \alpha^{(\ell)}$$

$$\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x}\|_2 \, \|\boldsymbol{x}'\|_2 \, g(\alpha^{(\ell)})$$

$$\alpha^{(\ell)} = \frac{\Sigma^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{\Sigma^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x})\Sigma^{(\ell-1)}(\boldsymbol{x}', \boldsymbol{x}')}}$$

$$g(x) = \frac{\sqrt{1 - x^2} + (\pi - \cos^{-1}(x))x}{\pi}$$

Efficient NTK Approximation

NTK with ReLU activation [ADH+19]**:** for layer $\ell = 1, \ldots, L$ and $\boldsymbol{x} \in \mathbb{R}^d$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

A key approach:

– If $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Lambda}^{(\ell)\prime} \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Psi}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)\prime} \rangle$

Efficient NTK Approximation

NTK with ReLU activation [ADH+19]**:** for layer $\ell = 1, \ldots, L$ and $\boldsymbol{x} \in \mathbb{R}^d$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

A key approach:

– If $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Lambda}^{(\ell)'} \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Psi}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)'} \rangle$

$$\begin{aligned} K_{\mathrm{NTK}}^{(1)}(\boldsymbol{x}, \boldsymbol{x}') &= K_{\mathrm{NTK}}^{(0)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \langle \boldsymbol{\Lambda}^{(1)}, \boldsymbol{\Lambda}^{(1)'} \rangle + \langle \boldsymbol{\Psi}^{(1)}, \boldsymbol{\Psi}^{(1)'} \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \cdot \langle \boldsymbol{\Lambda}^{(1)}, \boldsymbol{\Lambda}^{(1)'} \rangle + \langle \boldsymbol{\Psi}^{(1)}, \boldsymbol{\Psi}^{(1)'} \rangle \\ &= \left\langle [\boldsymbol{x} \otimes \boldsymbol{\Lambda}^{(1)}, \ \boldsymbol{\Psi}^{(1)}], [\boldsymbol{x}' \otimes \boldsymbol{\Lambda}^{(1)'}, \ \boldsymbol{\Psi}^{(1)'}] \right\rangle \end{aligned}$$

$\otimes$: tensor product

Efficient NTK Approximation

NTK with ReLU activation [ADH+19]**:** for layer $\ell = 1, \ldots, L$ and $\boldsymbol{x} \in \mathbb{R}^d$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

A key approach:

– If $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Lambda}^{(\ell)'} \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Psi}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)'} \rangle$

$$\boldsymbol{\Phi}^{(\ell)} = [\boldsymbol{\Phi}^{(\ell-1)} \otimes \boldsymbol{\Lambda}^{(\ell)}, \ \boldsymbol{\Psi}^{(\ell)}], \qquad \boldsymbol{\Phi}^{(0)} = \boldsymbol{x}$$

such that $K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Phi}^{(\ell)}, \boldsymbol{\Phi}^{(\ell)'} \rangle$

$\otimes$: tensor product

Efficient NTK Approximation

NTK with ReLU activation [ADH+19]**:** for layer $\ell = 1, \ldots, L$ and $\boldsymbol{x} \in \mathbb{R}^d$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

A key approach:

– If $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Lambda}^{(\ell)\prime} \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Psi}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)\prime} \rangle$

$$\boldsymbol{\Phi}^{(\ell)} = [\boldsymbol{\Phi}^{(\ell-1)} \otimes \boldsymbol{\Lambda}^{(\ell)}, \ \boldsymbol{\Psi}^{(\ell)}], \qquad \boldsymbol{\Phi}^{(0)} = \boldsymbol{x}$$

such that $K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Phi}^{(\ell)}, \boldsymbol{\Phi}^{(\ell)\prime} \rangle$

$\otimes$: tensor product

Q: How to find $\boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)}$ that approximates $\dot{\Sigma}^{(\ell)}, \Sigma^{(\ell)}$

$\Rightarrow$ Random Features [CS09], Polynomial Sketch [AKK+21]

NTK Random Features

Goal: $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Lambda}^{(\ell)'} \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Psi}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)'} \rangle \;\; \ell = 1, \dots, L$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

What is Random Features?

– A feature map $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that

$$f(\boldsymbol{x}, \boldsymbol{x}') = \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{P}} \left[\phi(\boldsymbol{x}, \boldsymbol{w}) \cdot \phi(\boldsymbol{x}', \boldsymbol{w}) \right]$$

– Sample m independent vectors $\boldsymbol{w}_1, \dots, \boldsymbol{w}_m$ and construct

$$\boldsymbol{\Phi} = \frac{1}{\sqrt{m}} \left[\phi(\boldsymbol{x}, \boldsymbol{w}_1), \dots, \phi(\boldsymbol{x}, \boldsymbol{w}_m) \right]^\top \in \mathbb{R}^m \qquad \text{(Random Features)}$$

such that $f(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[\langle \boldsymbol{\Phi}, \boldsymbol{\Phi}' \rangle]$

NTK Random Features

Goal: $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Lambda}^{(\ell)}, {\boldsymbol{\Lambda}^{(\ell)}}' \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Psi}^{(\ell)}, {\boldsymbol{\Psi}^{(\ell)}}' \rangle \;\; \ell = 1, \dots, L$

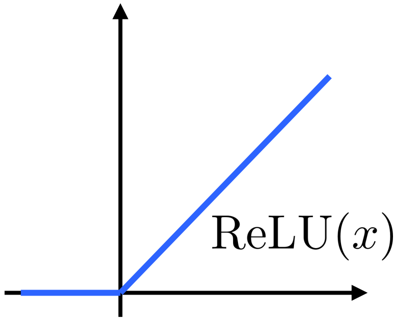
$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

Random Features of Arc-cosine Kernel [CS09]:

$$\Sigma^{(1)}(\boldsymbol{x}, \boldsymbol{x}') = 2 \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{N}} [\mathrm{ReLU}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) \cdot \mathrm{ReLU}(\langle \boldsymbol{w}, \boldsymbol{x}' \rangle)] = \mathbb{E} \left[\langle \boldsymbol{\Psi}, \boldsymbol{\Psi}' \rangle \right]$$

– m_1 independent samples:

$$\boldsymbol{\Psi} = \sqrt{\frac{2}{m_1}} \, \mathrm{ReLU} \left(\langle \boldsymbol{w}_1, \boldsymbol{x} \rangle, \dots, \langle \boldsymbol{w}_{m_1}, \boldsymbol{x} \rangle \right)^\top \in \mathbb{R}^{m_1}$$



NTK Random Features

Goal: $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Lambda}^{(\ell)}, {\boldsymbol{\Lambda}^{(\ell)}}' \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Psi}^{(\ell)}, {\boldsymbol{\Psi}^{(\ell)}}' \rangle \;\; \ell = 1, \ldots, L$

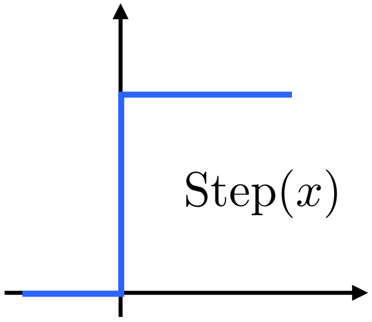
$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

Random Features of Arc-cosine Kernel [CS09]:

$$\dot{\Sigma}^{(1)}(\boldsymbol{x}, \boldsymbol{x}') = 2 \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{N}} [\mathrm{Step}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) \cdot \mathrm{Step}(\langle \boldsymbol{w}, \boldsymbol{x}' \rangle)] \qquad = \mathbb{E}\left[\langle \boldsymbol{\Lambda}, \boldsymbol{\Lambda}' \rangle\right]$$

– m_0 independent samples:

$$\boldsymbol{\Lambda} = \sqrt{\frac{2}{m_0}} \, \mathrm{Step}\left(\langle \boldsymbol{w}_1, \boldsymbol{x} \rangle, \ldots, \langle \boldsymbol{w}_{m_1}, \boldsymbol{x} \rangle\right)^\top \in \mathbb{R}^{m_0}$$



NTK Random Features

NTK Random Features.

Initialize $\boldsymbol{\Phi}^{(0)} = \boldsymbol{\Psi}^{(0)} = \boldsymbol{x} \in \mathbb{R}^d$

For $\ell = 1, \ldots, L$

Random Features

$$\boldsymbol{\Lambda}^{(\ell)} = \sqrt{\frac{2}{m_0}}\,\mathrm{Step}(\boldsymbol{W}'\boldsymbol{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_0} \qquad \boldsymbol{W}'_{ij} \sim \mathcal{N}(0,1)$$

$$\boldsymbol{\Psi}^{(\ell)} = \sqrt{\frac{2}{m_1}}\,\mathrm{ReLU}(\boldsymbol{W}\boldsymbol{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_1} \qquad \boldsymbol{W}_{ij} \sim \mathcal{N}(0,1)$$

$$\boldsymbol{\Phi}^{(\ell)} = [\ \boldsymbol{\Phi}^{(\ell-1)} \otimes \boldsymbol{\Lambda}^{(\ell)},\ \ \boldsymbol{\Psi}^{(\ell)}\ ]$$

Return $\boldsymbol{\Phi}^{(L)}$ such that $K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Phi}^{(L)}, \boldsymbol{\Phi}^{(L)'} \rangle$

NTK Random Features

NTK Random Features.

Initialize $\mathbf{\Phi}^{(0)} = \mathbf{\Psi}^{(0)} = \boldsymbol{x} \in \mathbb{R}^d$

For $\ell = 1, \ldots, L$

Random Features

$$\mathbf{\Lambda}^{(\ell)} = \sqrt{\frac{2}{m_0}}\, \mathrm{Step}(\boldsymbol{W}'\mathbf{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_0} \qquad \boldsymbol{W}'_{ij} \sim \mathcal{N}(0,1)$$

$$\mathbf{\Psi}^{(\ell)} = \sqrt{\frac{2}{m_1}}\, \mathrm{ReLU}(\boldsymbol{W}\mathbf{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_1} \qquad \boldsymbol{W}_{ij} \sim \mathcal{N}(0,1)$$

$$\mathbf{\Phi}^{(\ell)} = [\ \mathbf{\Phi}^{(\ell-1)} \otimes \mathbf{\Lambda}^{(\ell)},\ \ \mathbf{\Psi}^{(\ell)}\ ]$$

Return $\mathbf{\Phi}^{(L)}$ such that $K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \mathbf{\Phi}^{(L)}, \mathbf{\Phi}^{(L)'} \rangle$

Dimension of tensor product: $\dim(\boldsymbol{x} \otimes \boldsymbol{y}) = \dim(\boldsymbol{x}) \cdot \dim(\boldsymbol{y})$

Dimension of $\mathbf{\Phi}^{(L)}$ is $\mathcal{O}(m_0^L(m_1 + d))$ 😭

When L is large, the output feature can be very large ⇒ no efficiency

NTK Random Features

Computational bottleneck:

$$\mathbf{\Lambda}^{(\ell)} = \sqrt{\frac{2}{m_0}}\,\mathrm{Step}(\boldsymbol{W}'\mathbf{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_0} \qquad \boldsymbol{W}'_{ij} \sim \mathcal{N}(0,1)$$

$$\mathbf{\Psi}^{(\ell)} = \sqrt{\frac{2}{m_1}}\,\mathrm{ReLU}(\boldsymbol{W}\mathbf{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_1} \qquad \boldsymbol{W}_{ij} \sim \mathcal{N}(0,1)$$

$$\mathbf{\Phi}^{(\ell)} = \left[\ \mathbf{\Phi}^{(\ell-1)} \otimes \mathbf{\Lambda}^{(\ell)},\ \ \mathbf{\Psi}^{(\ell)}\ \right]$$

TensorSketch [PP13; AKK+20]:

$$\langle \boldsymbol{x} \otimes \boldsymbol{y}, \boldsymbol{z} \otimes \boldsymbol{w} \rangle = \mathbb{E}\left\langle \mathcal{T}(\boldsymbol{x},\boldsymbol{y}), \mathcal{T}(\boldsymbol{z},\boldsymbol{w}) \right\rangle, \qquad \boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w} \in \mathbb{R}^d$$

- Inner-product preserving dimensionality reduction
- $\mathcal{T}: \mathbb{R}^{d^2} \to \mathbb{R}^{m_\mathrm{s}}$ can be computed in time $\mathcal{O}(d + m_\mathrm{s} \log m_\mathrm{s})$ using FFT
- m_s is a tradeoff between runtime and error bound
- $m_\mathrm{s} = \mathcal{O}\left(\frac{1}{\varepsilon^2} \log^3 \frac{1}{\varepsilon\delta}\right)$ for given accuracy ε and failure probability δ

NTK Random Features

NTK Random Features.

Initialize $\boldsymbol{\Phi}^{(0)} = \boldsymbol{\Psi}^{(0)} = \boldsymbol{x} \in \mathbb{R}^d$

For $\ell = 1, \ldots, L$

Random Features

$$\boldsymbol{\Lambda}^{(\ell)} = \sqrt{\frac{2}{m_0}}\, \mathrm{Step}(\boldsymbol{W}'\boldsymbol{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_0} \qquad \boldsymbol{W}'_{ij} \sim \mathcal{N}(0,1)$$

$$\boldsymbol{\Psi}^{(\ell)} = \sqrt{\frac{2}{m_1}}\, \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{\Psi}^{(\ell-1)}) \in \mathbb{R}^{m_1} \qquad \boldsymbol{W}_{ij} \sim \mathcal{N}(0,1)$$

$$\boldsymbol{\Phi}^{(\ell)} = [\mathcal{T}(\boldsymbol{\Phi}^{(\ell-1)}, \boldsymbol{\Lambda}^{(\ell)}),\ \boldsymbol{\Psi}^{(\ell)}]$$

$\mathcal{T}$: TensorSketch to $\mathbb{R}^{m_s}$

Return $\boldsymbol{\Phi}^{(L)}$ such that $K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Phi}^{(L)}, {\boldsymbol{\Phi}^{(L)}}' \rangle$

Dimension of $\boldsymbol{\Phi}^{(L)}$ is $m_1 + m_s$ due to Tensor Sketch 😁

Larger m_1, m_0, m_s guarantee more accurate NTK approximation

Question: what is the error bound in terms of m_1, m_0, m_s ?

NTK Random Features

Theorem 1. Given $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and $\delta \in (0,1), \varepsilon \in (0, 1/L)$, if

$$m_0 = \mathcal{O}\left(\frac{L^2}{\varepsilon^2} \log \frac{L}{\delta}\right), \; m_1 = \mathcal{O}\left(\frac{L^6}{\varepsilon^4} \log \frac{L}{\delta}\right), \; m_{\mathbf{s}} = \mathcal{O}\left(\frac{L^2}{\varepsilon^2} \log^3 \frac{L}{\varepsilon\delta}\right),$$

then

$$\Pr\left[\left|\left\langle \boldsymbol{\Phi}^{(L)}, \boldsymbol{\Phi}^{(L)'} \right\rangle - K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \le (L+1)\varepsilon\right] \ge 1 - \delta$$

NTK Random Features

Theorem 1. Given $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and $\delta \in (0,1), \varepsilon \in (0, 1/L)$, if

$$m_0 = \mathcal{O}\left(\frac{L^2}{\varepsilon^2} \log \frac{L}{\delta}\right), \; m_1 = \mathcal{O}\left(\frac{L^6}{\varepsilon^4} \log \frac{L}{\delta}\right), \; m_{\mathrm{s}} = \mathcal{O}\left(\frac{L^2}{\varepsilon^2} \log^3 \frac{L}{\varepsilon\delta}\right),$$

then

$$\Pr\left[\left|\left\langle \boldsymbol{\Phi}^{(L)}, \boldsymbol{\Phi}^{(L)\prime} \right\rangle - K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \le (L+1)\varepsilon\right] \ge 1 - \delta$$

[ADH+19] showed that the gradient random features can guarantee

$$\Pr\left[\left|\left\langle \frac{\partial f(\boldsymbol{x}, \theta)}{\partial \theta}, \frac{\partial f(\boldsymbol{x}', \theta)}{\partial \theta} \right\rangle - K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \le (L+1)\varepsilon\right] \ge 1 - \delta$$

– Their feature dimension can be $\|\theta\|_0 = \widetilde{\Omega}\left(\frac{L^{13}}{\varepsilon^8}\right)$

– Our feature dimension is $m_1 + m_{\mathrm{s}} = \widetilde{\mathcal{O}}\left(\frac{L^6}{\varepsilon^4}\right)$ 😁

NTK Random Features

Theorem 1. Given $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and $\delta \in (0,1), \varepsilon \in (0, 1/L)$, if

$$m_0 = \mathcal{O}\left(\frac{L^2}{\varepsilon^2}\log\frac{L}{\delta}\right), \; m_1 = \mathcal{O}\left(\frac{L^6}{\varepsilon^4}\log\frac{L}{\delta}\right), \; m_{\mathrm{s}} = \mathcal{O}\left(\frac{L^2}{\varepsilon^2}\log^3\frac{L}{\varepsilon\delta}\right),$$

then

$$\Pr\left[\left|\left\langle \boldsymbol{\Phi}^{(L)}, \boldsymbol{\Phi}^{(L)\prime}\right\rangle - K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \le (L+1)\varepsilon\right] \ge 1 - \delta$$

[ADH+19] showed that the gradient random features can guarantee

$$\Pr\left[\left|\left\langle \frac{\partial f(\boldsymbol{x}, \theta)}{\partial \theta}, \frac{\partial f(\boldsymbol{x}', \theta)}{\partial \theta}\right\rangle - K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \le (L+1)\varepsilon\right] \ge 1 - \delta$$

- Their feature dimension can be $\|\theta\|_0 = \widetilde{\Omega}\left(\frac{L^{13}}{\varepsilon^8}\right)$
- Our feature dimension is $m_1 + m_{\mathrm{s}} = \widetilde{\mathcal{O}}\left(\frac{L^6}{\varepsilon^4}\right)$ 😁

We can improve the error bound using importance sampling

NTK Sketch

Goal: $\dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Lambda}^{(\ell)}, \boldsymbol{\Lambda}^{(\ell)'} \rangle$ and $\Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Psi}^{(\ell)}, \boldsymbol{\Psi}^{(\ell)'} \rangle$ $\ell = 1, \dots, L$

$$K_{\mathrm{NTK}}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') = K_{\mathrm{NTK}}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \dot{\Sigma}^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}') + \Sigma^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$$

PolySketch [AKK+20]:

- $\dot{\Sigma}^{(\ell)}, \Sigma^{(\ell)}$ are dot-product kernels & approximated by a polynomial

$$\Sigma^{(1)}(\boldsymbol{x}, \boldsymbol{x}') = 1 - \frac{1}{\pi} \cos^{-1} \alpha \approx \sum_{j=0}^{p} c_j \alpha^j \qquad \alpha := \frac{\langle \boldsymbol{x}, \boldsymbol{x}' \rangle}{\|\boldsymbol{x}\|_2 \|\boldsymbol{x}'\|_2}$$

- Each monomial term can be approximated by TensorSketch [PP13]

$$\alpha^j \approx \langle \mathcal{T}(\boldsymbol{x}, j), \mathcal{T}(\boldsymbol{x}', j) \rangle$$

$\mathcal{T}(\cdot, j)$: TensorSketch of degree *j*

$$\boldsymbol{\Lambda}^{(1)} \approx \left[\sqrt{c_0}, \sqrt{c_1} \mathcal{T}(\boldsymbol{x}, 1), \dots, \sqrt{c_p} \mathcal{T}(\boldsymbol{x}, p) \right]$$

$$\Sigma^{(1)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \boldsymbol{\Lambda}^{(1)}, \boldsymbol{\Lambda}^{(1)'} \rangle$$

NTK Sketch

NTK Sketch.

Initialize $\mathbf{\Phi}^{(0)} = \mathbf{\Lambda}^{(0)} = \boldsymbol{x} \in \mathbb{R}^d$

For $\ell = 1, \ldots, L$

Sketching method

$$\mathbf{\Lambda}^{(\ell)} = \mathrm{PolySketch}(\mathbf{\Psi}^{(\ell-1)}, p) \text{ for } 1 - \frac{1}{\pi}\cos^{-1}(x)$$

$$\mathbf{\Psi}^{(\ell)} = \mathrm{PolySketch}(\mathbf{\Psi}^{(\ell-1)}, p') \text{ for } \frac{1}{\pi}\left(\sqrt{1-x^2} + (\pi - \cos^{-1}(x))x\right)$$

$$\mathbf{\Phi}^{(\ell)} = [\mathcal{T}(\mathbf{\Phi}^{(\ell-1)}, \mathbf{\Lambda}^{(\ell)}),\ \mathbf{\Psi}^{(\ell)}]$$

$\mathcal{T}$: TensorSketch to $\mathbb{R}^{m_s}$

Return $\mathbf{\Phi}^{(L)}$ such that $K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \mathbf{\Phi}^{(L)}, \mathbf{\Phi}^{(L)'} \rangle$

$\dot{\Sigma}^{(\ell)}, \Sigma^{(\ell)}$ can be approximated by PolySketch [AKK+20]

PolySketch is faster than Random Features (i.e., matrix-matrix products)

With degree p and sketch dimension m , it can run in $\mathcal{O}(p(d + m\log m))$

NTK Sketch

Theorem 2. Given $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and $\delta \in (0,1), \varepsilon \in (0, 1/L)$, the NTK Sketch computes $\boldsymbol{\Phi}^{(L)} \in \mathbb{R}^m, m = \mathcal{O}(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ in time

$$\widetilde{\mathcal{O}}\left(\frac{L^{11}}{\varepsilon^{6.7}} + \frac{L^3}{\varepsilon^2} d\right)$$

such that

$$\Pr\left[\left|\left\langle \boldsymbol{\Phi}^{(L)}, \boldsymbol{\Phi}^{(L)'} \right\rangle - K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \le \varepsilon \cdot K_{\mathrm{NTK}}^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right] \ge 1 - \delta$$

Feature dimension m does not depend on L

Compared to NTK Random Features, NTK Sketch is fast and efficient

	Feature dimension	Running time
NTK Random Features	$\widetilde{\mathcal{O}}\left(\frac{L^6}{\varepsilon^4}\right)$	$\widetilde{\mathcal{O}}\left(\frac{L^{13}}{\varepsilon^8}\right)$

NTK RF/Sketch methods can be extended to Convolutional Neural Networks (CNNs)

Outline

Introduction
- Kernel Regression
- Neural Tangent Kernel
- Existing Work

NTK Feature Map Construction
- Exact NTK Computation
- NTK Random Features
- NTK Sketch

Experiments
- Classification on MNIST datasets
- Classification on CIFAR-10 datasets

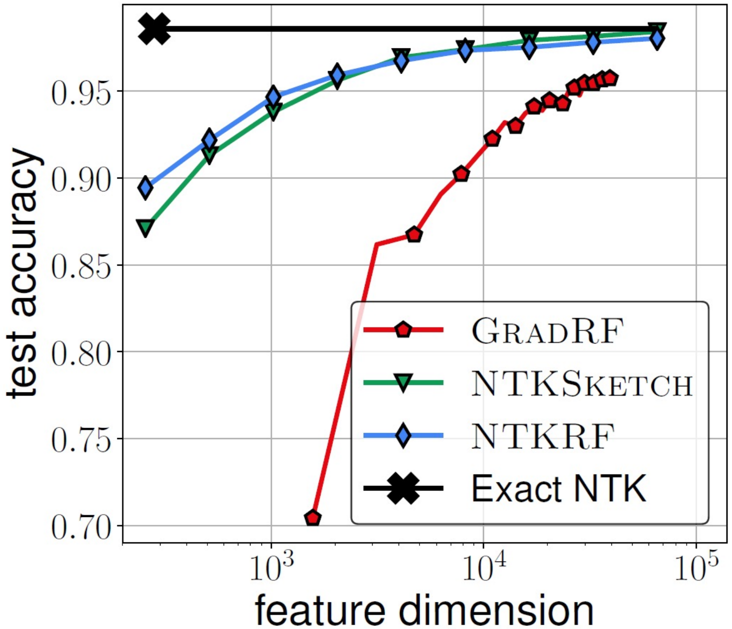
Conclusion

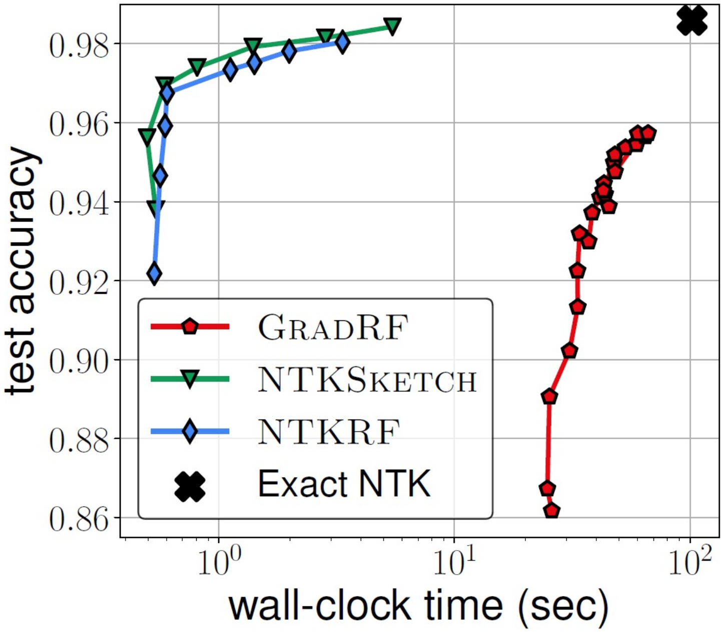
Experiments: MNIST Classification

Comparison to other NTK approximation methods:

- Kernel regression by the exact NTK (Exact NTK)
- Gradient features by Monte-Carlo sampling (GradRF)
- NTK Random Features (NTKRF) / Sketch (NTKSketch)

Test accuracy versus feature dimension / computing time ($L = 1$)



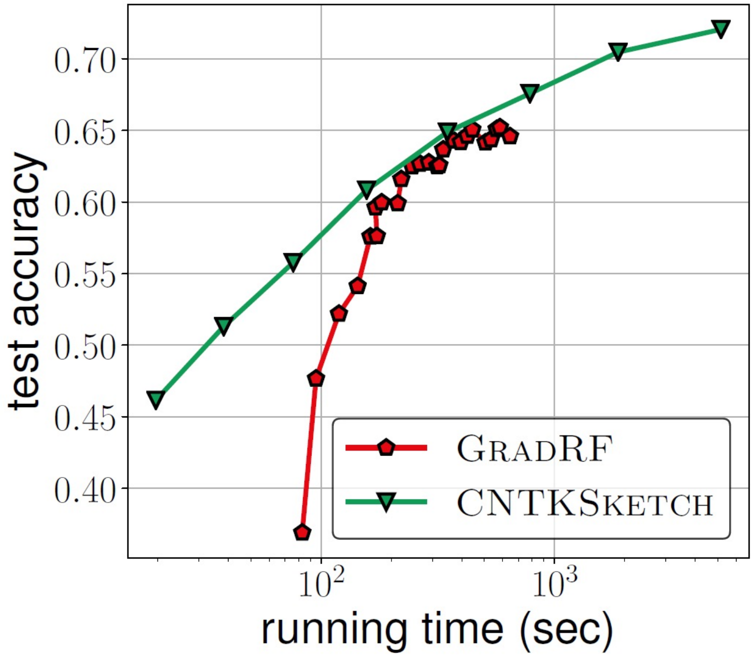


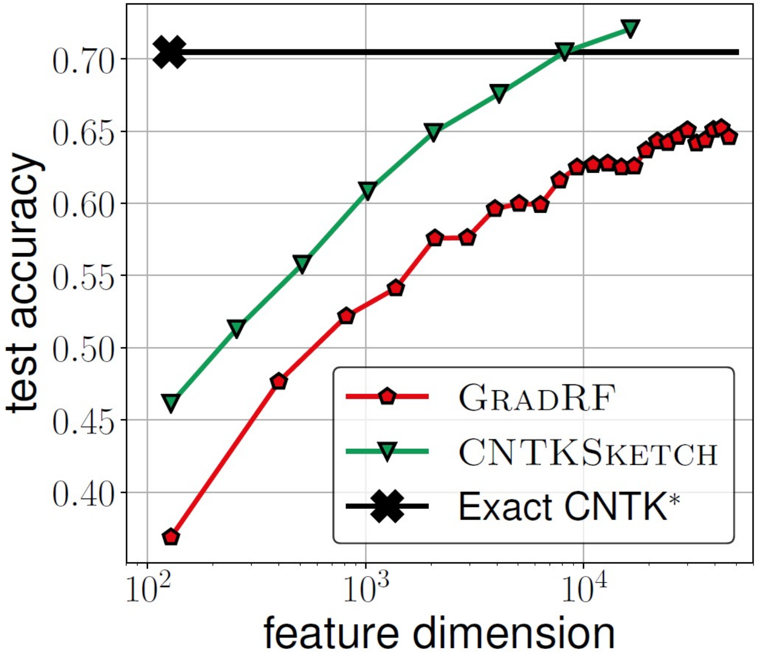
Experiments: CIFAR-10 Classification

Comparison to other NTK approximation methods:

- Kernel regression by the exact CNTK (Exact CNTK)
- Gradient features by Monte-Carlo sampling (GradRF)
- CNTK Sketch (CNTKSketch)

Test accuracy versus feature dimension / computing time ($L = 3$)





Experiments: CIFAR-10 Classification

Comparison to other NTK approximation methods:

- Kernel regression by the exact CNTK (Exact CNTK)
- Gradient features by Monte-Carlo sampling (GradRF)
- CNTK Sketch (CNTKSketch)

Test accuracy versus feature dimension / computing time ($L = 3$)

	CNTKSketch (ours)			GradRF			Exact CNTK	CNN
Feature dimension	4,096	8,192	16,384	9,328	17,040	42,816		
Test accuracy (%)	67.58	70.46	72.06	62.49	62.57	65.21	70.47*	63.81*
Time (s)	780	1,870	5,160	300	360	580	> 1,000,000	

* means that the result is from [ADH+19]

Conclusion

Summary:

- We propose efficient feature maps of NTK / CNTK
- We design two approaches utilize sketching algorithm and arc-cosine random features, respectively
- We provide an entry-wise error bound for both algorithms
- We additionally provide a spectral approximation bound of random features approach with leverage score sampling
- The proposed methods outperform other baselines

Future work:

- Spectral approximation guarantee for deeper layers
- Applying our method to convolutional NTK, attention network, etc