When do neural networks outperform kernel methods?
Are kernel methods doomed?

Theodor Misiakiewicz

Stanford University

June 21st, 2022

ELLIS reading group on Mathematics of Deep Learning

Joint work with Behrooz Ghorbani (Google Research), Song Mei (UC Berkeley) and Andrea Montanari (Stanford)

- Two mysteries of deep learning: **optimization** and **generalization**.
- Connection between neural networks (NNs) and kernel machines:
  - Not new [Neal, 1996], [Cho, Saul, 2009].
  - Large-width NNs trained in some regime $\approx$ kernel methods:
    the **Neural Tangent Kernel** [Jacot et al., 2018].
- Kernel methods: regime of NNs where optimization is easy and we can focus on generalization (attractive to theoricians).
- How interesting is this kernel regime to explain NNs performance?
  Can be seen as a first order approximations of NNs:
  - Theoretical separation results in specific cases [Daniely, Malach, 2020]
  - Kernels perform as well as NNs on some datasets [Arora et al., 2019]
  - NTKs perform much better than previous handcrafted kernels.
- In this talk, I will show how data structure plays an important role in inducing these performance gaps.

Random Features (RF) and Neural Tangent (NT) model

- Neural network: $\mathrm{NN}_p(\boldsymbol{x};\boldsymbol{\theta})$, $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{\theta} \in \mathbb{R}^p$.
  e.g., fully-connected neural network:

$$\mathrm{NN}_p(\boldsymbol{x};\boldsymbol{\theta}) = \langle \boldsymbol{a}, \sigma(\dots \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x}))\rangle.$$

- Neural Tangent (NT) model: linear approximation of NN_p around initialization $\boldsymbol{\theta}_0$

$$\mathrm{NT}_p(\boldsymbol{x};\boldsymbol{\beta},\boldsymbol{\theta}_0) = \langle \boldsymbol{\beta}, \nabla_{\boldsymbol{\theta}}\mathrm{NN}_p(\boldsymbol{x};\boldsymbol{\theta}_0)\rangle.$$

- Limiting kernel ($p \to \infty$):

$$K_{\mathrm{NT}}(\boldsymbol{x},\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\theta}_0}[\langle \nabla_{\boldsymbol{\theta}}\mathrm{NN}_p(\boldsymbol{x};\boldsymbol{\theta}_0), \nabla_{\boldsymbol{\theta}}\mathrm{NN}_p(\boldsymbol{y};\boldsymbol{\theta}_0)\rangle].$$

GD trained NNs $\approx$ NT model in some regime

Coupled gradient descent on empirical squared loss:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\theta}^t = -\nabla_{\boldsymbol{\theta}}\hat{\mathbb{E}}[(y - \mathsf{NN}_p(\boldsymbol{x};\boldsymbol{\theta}^t))^2], \qquad \boldsymbol{\theta}^0 = \boldsymbol{\theta}_0,$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\beta}^t = -\nabla_{\boldsymbol{\beta}}\hat{\mathbb{E}}[(y - \mathsf{NT}_p(\boldsymbol{x};\boldsymbol{\beta}^t,\boldsymbol{\theta}_0))^2], \qquad \boldsymbol{\beta}^0 = 0.$$

Theorem (informal)

For any $\varepsilon > 0$, for large enough NNs and proper random initialization $\boldsymbol{\theta}_0$ (Xavier initialization), we have with high probability

$$\sup_{t\geq 0}\ \sup_{\|\boldsymbol{x}\|_2=1} |\mathsf{NN}_p(\boldsymbol{x};\boldsymbol{\theta}^t) - \mathsf{NT}_p(\boldsymbol{x};\boldsymbol{\beta}^t,\boldsymbol{\theta}_0)| \leq \varepsilon.$$

$\Longrightarrow$ Intuition: *weights do not change much* and $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}_0\|_2$ remains small.

[Jacot, Gabriel, Hongler,'18] , [Du, Zhai, Poczos, Singh,'18], [Du, Lee, Li, Wang, Zhai,'18], [Allen-Zhu, Li, Song,'18] , [Zou, Cao, Zhou, Gu,'18], [Oymak, Soltanolkotabi,'18], [Chizat, Oyallon, Bach,'19], ...

The ‘Kernel Regime’

- Optimization success: explain the "optimization mystery".
- . . . but what about generalization?
  - **Lazy Training:** weights hardly change, there is ‘no feature learning’. [Chizat et al.,’18]
  - Empirically, NNs perform often better than their linearized counterparts.
- "Kernel methods do not explain the good performance of NNs". More nuanced:
  - NTK achieves near state-of-the-art results on UCI dataset [Arora et al.,’19].
  - Performance gap between NTKs and NNs depend on the dataset and architecture [Geiger et al.,’20].

New approach for kernel engineering

- Progress on CIFAR-10:

Paper	Method	Accuracy
[Coates, Lee, Ng,'11]	feature learning + linear regression	77%
[Arora et al.,'19]	CNTK (data independent)	77%
[Li et al.,'19]	CRFK + data dependent preprocessing	89%
[Shankar et al.,'20]	CRFK (data independent)	90%
[Bietti,'21]	Simple CK (data independent)	89%
-	CNN	$> 99\%$

- Unexpected good performance.

Questions

- Why are kernel methods not as good as NNs in general?
- When can we expect kernel methods to perform well, comparable to NNs?
- Can we quantify the benefits of using convolutional kernels against inner-product kernels (e.g., Gaussian kernel)?

Outline of the talk

Three 'stylized' scenarios to understand the gap between NNs and kernel methods:

A. **Isotropic covariates model:**
   - large gap between NN and NT.

B. **Spiked covariates model:**
   - smaller gap between NN and NT.

C. **Local and invariant function estimation (with isotropic covariates):**
   - quantify the benefits of convolutional kernels over standard kernels.

A. Isotropic covariates model

Two-layers fully-connected neural networks

$$\mathrm{NN}_N(\boldsymbol{x}; \boldsymbol{\Theta}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle), \qquad a_i \in \mathbb{R}, \boldsymbol{w}_i \in \mathbb{R}^d.$$

- As NT model, we use the following toy model ($\boldsymbol{W} = (\boldsymbol{w}_i)_{i \in [N]} \sim_{iid} \mathrm{Unif}(\mathbb{S}^{d-1})$ fixed):

$$\mathrm{RF}_N(\boldsymbol{x}; a, \boldsymbol{W}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle), \qquad a_i \in \mathbb{R},$$

  (linearization of top layer weight)

- Also called the Random Feature model [Rahimi,Recht,'08].

- Qualitatively (quantitatively in some cases) captures behavior of more general NT models.

Data isotropic on the sphere

- We consider data uniformly distributed on the sphere:

$$\mathbb{S}^{d-1}(\sqrt{d}) = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_2 = \sqrt{d}\}.$$

- Given n iid samples $\{(y_i, \boldsymbol{x}_i)\}_{i \in [n]}$ with target function f_*:

$$y_i = f_\star(\boldsymbol{x}_i) + \varepsilon_i,$$

$$\boldsymbol{x}_i \sim_{iid} \mathrm{Unif}(\mathbb{S}^{d-1}(\sqrt{d})), \qquad \varepsilon_i \sim_{iid} N(0, \tau^2).$$

- Ridge regression with RF model:

$$\hat{\boldsymbol{a}}(\lambda) = \arg\min_{\boldsymbol{a}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{N} a_j \sigma(\langle \boldsymbol{w}_j, \boldsymbol{x}_i \rangle) \Big)^2 + \lambda \|\boldsymbol{a}\|_2^2 \right\}.$$

Denote the solution $\widehat{\mathrm{RF}}_{n,N,\lambda} = \mathrm{RF}_N(\cdot; \hat{\boldsymbol{a}}(\lambda), \boldsymbol{W})$.

Test error of RF model

Theorem (Mei, Misiakiewicz, Montanari, 2021)

Assume $d^{\ell+\delta} \leq \min(n, N) \leq d^{\ell+1-\delta}$ and $\max\left(\frac{n}{N}, \frac{N}{n}\right) \geq d^{\delta}$. Then,

$$\text{Test error} = \| f_\star - \widehat{\mathrm{RF}}_{n,N,\lambda} \|_{L^2}^2 = \| \mathrm{P}_{>\ell} f_\star \|_{L^2}^2 + o_d(1).$$

$\mathrm{P}_{>\ell}$: projection orthogonal to the space of degree-ℓ polynomials.

- For d^ℓ parameters and samples, RF fits the best degree-ℓ polynomial approximation, and nothing else!

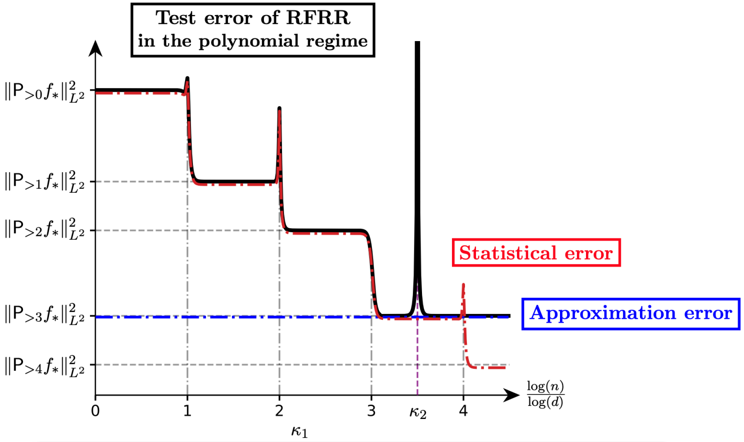
- For $N \to \infty$, this correspond to a kernel method with inner-product kernel.

  For NTK of fully-connected NNs of any depth: to get a degree-ℓ polynomial approximation to the target function, needs $n = d^\ell$ samples (no effect of depth).

- **Curse of dimensionality!!!**

The truncated staircase decay

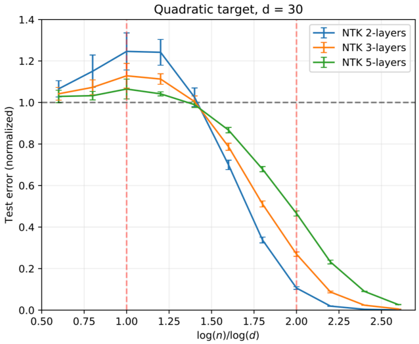
For $n = d^{\kappa_1}$ and $N = d^{\kappa_2}$, the test error is given by:



$$\text{Test error}(n, N) \approx \max\left\{\text{approx. error } (n = \infty, N), \text{ statistical error } (n, N = \infty)\right\}.$$

Numerical simulations

$$f_{\text{quad}} = \sum_{i \le \lfloor d/2 \rfloor} x_i^2 - \sum_{i > \lfloor d/2 \rfloor} x_i^2, \qquad f_{\text{cube}} = \sum_{i=1}^{d} (x_i^3 - 3x_i).$$



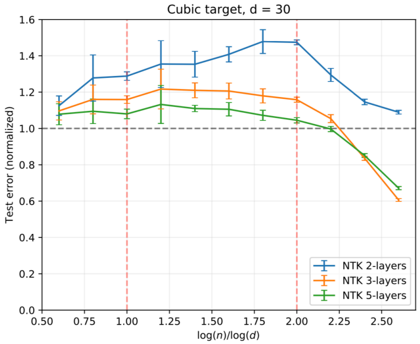


Figure: Test error of KRR with isotropic data for NTKs of different depth.

Performance gap between NN and kernel methods

- $n, N = d^{\ell}$ is necessary to learn in "worst case" all polynomials of degree-ℓ. So in "worst case", NNs can't outperform Kernel methods.

- However: we expect NNs to perform way better on 'low complexity' functions

  E.g., $f_\star(\boldsymbol{x}) = \sigma(\langle \boldsymbol{w}_\star, \boldsymbol{x}\rangle)$: if $N \propto 1$ and $n \propto d$, then GD solution:

  $$\| f_\star - \widehat{\mathrm{NN}}_{N,n} \|_{L^2}^2 = o_d(1) \qquad \text{[Bai et al.,'16]}$$

  while for RF, for any $\min(N, n) = d^{\ell}$:

  $$\| f_\star - \widehat{\mathrm{RF}}_{N,n} \|_{L^2}^2 = \| \mathrm{P}_{>\ell} f_\star \|_{L^2}^2 + o_d(1).$$

- More generally we expect NNs to vastly outperform kernel methods for target functions that only depend on a low dimensional subspace of the data [Bach,'17].

Intuition: fit $\varphi(x_1)$ using features $\sigma(\langle \boldsymbol{w}_i, \boldsymbol{x}\rangle)$

- For RF, $\boldsymbol{w}_i$'s are random, $\mathrm{corr}(x_1 \boldsymbol{e}_1, \langle \boldsymbol{w}_i, \boldsymbol{x}\rangle)^2 = \langle \boldsymbol{w}_i, \boldsymbol{e}_1\rangle^2 / \|\boldsymbol{w}_i\|_2^2$ is small in HD.
- For NN, $\boldsymbol{w}_i$'s can be chosen to have a large correlation with $\boldsymbol{e}_1$.

NN can 'adaptively learn' good $\boldsymbol{w}_i$'s while RF cannot.

In the isotropic covariates model:

- RF and standard kernel methods suffer from the **curse of dimensionality**:

  For $\min(N, n) \propto d^{\ell}$, only learn a degree-$\ell$ polynomial approximation.

- For target functions that depend on a **low dimensional projection** of the data, NNs can **adaptively learn a good representation** of the data and vastly outperform kernel methods.

B. Spiked covariates model

What about in practice?

- Some recent empirical studies showed that for some image classification tasks, NTK perform almost as well as fully-connected NNs [Geiger et al.,'20].
- For isotropic data, NTK shouldn't be able to learn much because of curse of dimensionality.
- However, for **real image datasets:**
  (1) Target function (labels) depends predominantly on the low frequency components of the images;
  (2) The images themselves have most of their spectrum concentrated on low frequencies.
- Real images: **highly anisotropic**.

We introduce a stylized setting: the *'spiked covariates model'*.

Spiked covariates model

Covariate vector: orthogonal matrix $[\boldsymbol{U}, \boldsymbol{U}^{\perp}]$

$$\boldsymbol{x} = \boldsymbol{U}\boldsymbol{z}_1 + \boldsymbol{U}^{\perp}\boldsymbol{z}_2, \qquad \boldsymbol{z}_1 \in \mathbb{R}^{d_s}, \boldsymbol{z}_2 \in \mathbb{R}^{d-d_s}.$$

Signal part: $\boldsymbol{z}_1 \sim \text{Unif}\left(\mathbb{S}^{d_s-1}(\sqrt{\text{snr}_c \cdot d_s})\right)$.

Noise part: $\boldsymbol{z}_2 \sim \text{Unif}\left(\mathbb{S}^{d-d_s-1}(\sqrt{d-d_s})\right)$

d_s = signal dimension.

snr_c = covariate SNR

Target function: $f_\star(\boldsymbol{x}) = \varphi(\boldsymbol{z}_1)$.

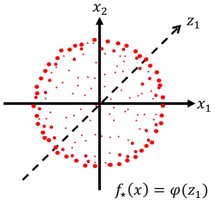


Figure: Isotropic covariates: $\text{snr}_c = 1$.

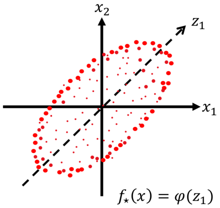


Figure: Anisotropic covariates: $\text{snr}_c > 1$.

Test error of KRR in the spiked covariates model

- Given iid samples $(\{\boldsymbol{x}_i, y_i\})_{i\in[n]}$ from the spiked covariates model.
- Effective dimension: $d_{\text{eff}} = \max(d_s, d/\text{snr}_c)$.

Theorem (Ghorbani, Mei, Misiakiewicz, Montanari, 2020)

Assume ${d_{\text{eff}}}^{\ell+\delta} \le \min(n, N) \le {d_{\text{eff}}}^{\ell+1-\delta}$ and $\max\left(\frac{n}{N}, \frac{N}{n}\right) \ge d^{\delta}$. Then,

$$\text{Test error} = \|f_\star - \widehat{\text{RF}}_{n,N,\lambda}\|_{L^2}^2 = \|\text{P}_{>\ell} f_\star\|_{L^2}^2 + o_d(1).$$

- For isotropic data, $\text{snr}_c = 1$ and $d_{\text{eff}} = d$, and we recover the previous theorem.
- As snr_c increases, $d_{\text{eff}} = d/\text{snr}_c$ decreases and we expect RF model to perform better and better.
- We expect NNs to learn features $\boldsymbol{w}_i$ aligned with $\boldsymbol{z}_1$ and to depend mildly on snr_c, and only depend on d_s.

NNs vs kernel methods in the spiked covariates model

Effective dimension: $d_{\text{eff}} = \max(d_s, d/\text{snr}_c)$.

To fit a degree-ℓ polynomial in $\boldsymbol{z}_1$:

- Kernel methods need $\Theta({d_{\text{eff}}}^{\ell})$ samples.
- NNs need $O({d_s}^{\ell})$ samples.

- $d_s \leq d_{\text{eff}} \leq d$:

  Test error: NN $\leq$ Kernel methods

- d_{eff} decreases with snr_c:
  - Isotropic features: $d_{\text{eff}} = d$ (low covariate SNR)

    Test error: NN $\ll$ Kernel methods.

  - Highly anisotropic features: $d_{\text{eff}} = d_s$ (high covariate SNR)

    Test error: NN $\sim$ Kernel methods.

Intuition for the gap: anisotropic case

Goal: fitting $\varphi(x_1)$ using features $\sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$.

Consider $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\mathsf{T}] = \mathrm{diag}(\mathsf{snr}_c, \mathrm{Id}_{d-1})$.

- To fit $\varphi(x_1)$, we need to use features $\sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$ that are correlated to x_1, i.e.,

$$\mathrm{corr}(x_1 \boldsymbol{e}_1, \langle \boldsymbol{w}_i, \boldsymbol{x} \rangle) = \mathsf{snr}_c \cdot \frac{\langle \boldsymbol{w}_i, \boldsymbol{e}_1 \rangle^2}{\| \boldsymbol{\Sigma}^{1/2} \boldsymbol{w}_i \|_2^2} = \mathsf{snr}_c \cdot O_{d,\mathbb{P}}(d^{-1}) = O_{d,\mathbb{P}}({d_{\mathrm{eff}}}^{-1}).$$

- For RF, when snr_c is large, all random $\boldsymbol{w}_i$'s are good.

- For NN, $\boldsymbol{w}_i$'s can be chosen to have a large correlation with $\boldsymbol{e}_1$.

RF/NT automatically have access to good $\boldsymbol{w}_i$'s.

Insight

Insight

Lower covariate SNR (data more isotropic) should lead to larger generalization gap between NNs and kernel methods.

How to test this insight?

In *image classification*, we expect

- Images to have most of their spectrum concentrated on low-frequencies ($\boldsymbol{z}_1$ part);
- The labels to depend predominantly on the low-frequencies ($y = f_\star(\boldsymbol{x}) = \varphi(\boldsymbol{z}_1)$).

Experiment: add noise to the high frequency components ($\boldsymbol{z}_2$ part) of the images (i.e., decreasing the snr_c).

Adding noise in covariates should increase the generalization gap between NN and kernel methods.

Numerical simulations

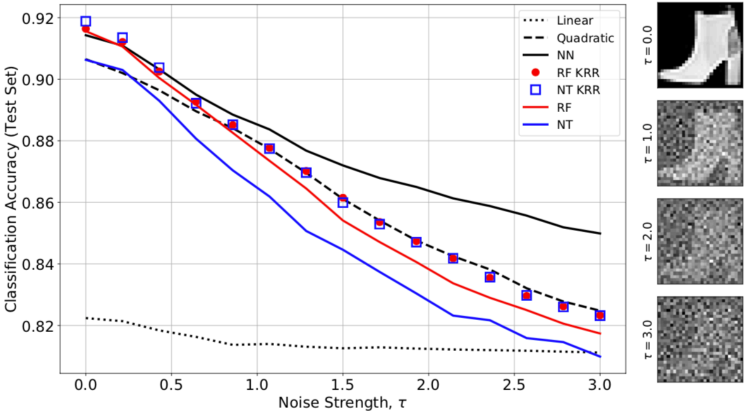


Figure: Test accuracy on FMNIST: adding noise to the high frequency components.

Spiked covariates model:
A controlling parameter of the performance gap between NN and kernel methods is

$$\mathsf{snr}_c = \text{Covariate SNR} = \frac{\text{Signal covariates variance}}{\text{Noise covariates variance}}.$$

- Small snr_c: large separation.
- Large snr_c: kernel methods perform closer to NN.

Intuition:

- When snr_c is small, RF/NT fail to find the signal covariates.
- When snr_c is big, RF/NT automatically find the signal covariates.
- NNs always look for the signal covariates.

C. Local and invariant function estimation on the hypercube

Natural structure in the target function

- In many learning tasks, the target function present some **natural properties**.

  E.g., image classification: labels **invariant under translation** of the images
  + depends on local information of the images.

- Effectiveness of NNs architectures is often loosely attributed to their ability to encode various structures present in the data.

  E.g., **Convolutions for locality and pooling for translation invariance.**

- CNNs perform better than fully-connected networks.

  Convolutional kernels perform better than inner-product kernels.

- **Can we quantify this gain?**

1-layer convolutional kernels

- Data $\boldsymbol{x} \sim \mathrm{Unif}(\{+1, -1\}^d)$. Consider patches of size $1 \leq q \leq d$:

$$\boldsymbol{x}_{(k)} = (x_k, x_{k+1}, \ldots, x_{k+q-1}).$$

- Consider a two-layer convolutional NN (local average pooling on segment size ω):

$$f_{CNN}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{\Theta}) = \sum_{i \in [N]} \sum_{k \in [d]} a_{ik} \sum_{s \in [\omega]} \sigma\left(\langle \boldsymbol{w}_i, \boldsymbol{x}_{(k+s)} \rangle\right)$$

- Associated NTK:

$$H^{CK}_{q,\omega}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k \in [d]} \sum_{s, s' \in [\omega]} h\big(\langle \boldsymbol{x}_{(k+s)}, \boldsymbol{y}_{(k+s')} \rangle / q\big)$$

- E.g., $q = d$ and $\omega = 1$,

$$H^{CK}_{d,1}(\boldsymbol{x}, \boldsymbol{y}) = H^{FC}(\boldsymbol{x}, \boldsymbol{y}) = h(\langle \boldsymbol{x}, \boldsymbol{y} \rangle / d)\,.$$

  This is a standard inner-product kernel (the NTK of fully-connected NNs).

Target function class

- No average pooling ($\omega = 1$):

$$H^{CK}_{q,1}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k \in [d]} h\big(\langle \boldsymbol{x}_{(k)}, \boldsymbol{y}_{(k)} \rangle / q\big) .$$

  Can learn q-local functions:

$$\mathrm{Loc}_q = \Big\{ f(\boldsymbol{x}) = \sum_{k \in [d]} g_k(\boldsymbol{x}_{(k)}) \Big\} .$$

- With global pooling ($\omega = d$):

$$H^{CK}_{q,d}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k, k' \in [d]} h\big(\langle \boldsymbol{x}_{(k)}, \boldsymbol{y}_{(k')} \rangle / q\big) .$$

  Can learn q-local cyclic-invariant functions:

$$\mathrm{CycLoc}_q = \Big\{ f(\boldsymbol{x}) = \sum_{k \in [d]} g(\boldsymbol{x}_{(k)}) \Big\} .$$

Benefits of having an adapted kernel

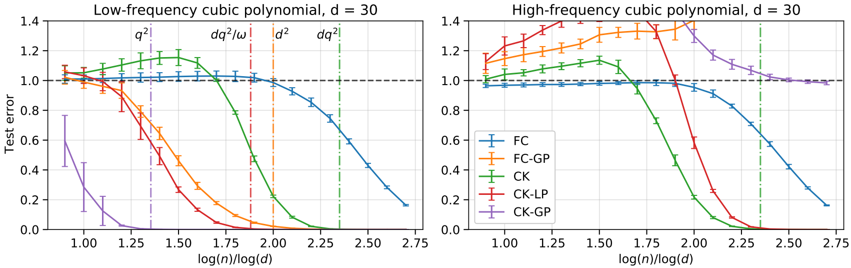
To learn a degree-ℓ polynomial that is q-local and cyclic invariant:

To fit a degree ℓ polynomial	$H_{d,1}^{CK}$	$H_{d,d}^{CK}$	$H_{q,1}^{CK}$	$H_{q,\omega}^{CK}$	$H_{q,d}^{CK}$
Sample size n	d^{ℓ}	$d^{\ell-1}$	$dq^{\ell-1}$	$dq^{\ell-1}/\omega$	$q^{\ell-1}$

- Convolutional kernels trade off approximation power for generalization power.
- Big sample efficiency gain by using a kernel that is adapted to the target function.
- 1-layer CK: 81% on CIFAR-10 [Bietti, 2021].
- 3-layer CK when properly trained $\approx$ 89% on CIFAR-10 (comparable to AlexNet).

Numerical simulations

$$f_{LF,3}(\boldsymbol{x}) = \frac{1}{\sqrt{d}} \sum_{i\in[d]} x_i x_{i+1} x_{i+2}\,, \qquad f_{HF,3}(\boldsymbol{x}) = \frac{1}{\sqrt{d}} \sum_{i\in[d]} (-1)^i \cdot x_i x_{i+1} x_{i+2}\,.$$



Summary

What I illustrated in this talk:

- **Isotropic covariates model:**
  Standard (inner-product) kernel methods suffer from the *curse of dimensionality* and perform inferior to NNs due to **lack of feature learning.**

- **Spiked covariates model:**
  For large covariate SNR (and low-dimensional data), **feature learning is unecessary** and kernel methods can also perform well.

- **Structured function estimation:**
  If the kernel includes some information on the target function (locality, invariance), it can perform much better.

Are kernels doomed to be inferior to NNs? Can we construct kernels that can adapt to data and match the performance of NNs?

Applications: robustness, reliability, interpretability, ...

References

1. *Linearized two-layers neural networks in high dimension*. Ghorbani, Mei, **M.**, Montanari (2019).

2. *When do neural networks outperform kernel methods?* Ghorbani, Mei, **M.**, Montanari (2020).

3. *Generalization error of random feature and kernel methods: Hypercontractivity and kernel matrix concentration*. Mei, **M.**, Montanari (2021).

4. *Learning with invariances in random features and kernel models*. Mei, **M.**, Montanari (2021).

5. *Learning with convolution and pooling operations in kernel methods*. **M.**, Mei (2021).