Numerical Understanding Of Neural Networks: From Representation to Learning Dynamics

Hongkai Zhao
Duke University

Joint work with Shijung Zhang, Haomin Zhou and Yimin Zhong

Research partially supported by NSF DMS-2012860, DMS-2309551.

Happy Birthday, Russ!

The goals and key ingredients in scientific computing

Approximate a mapping/function.

The goals and key ingredients in scientific computing

Approximate a mapping/function.

Key ingredients

- Representation: the starting point.

The goals and key ingredients in scientific computing

Approximate a mapping/function.

Key ingredients

- Representation: the starting point.
- Error (lost) function.

The goals and key ingredients in scientific computing

Approximate a mapping/function.

Key ingredients

- Representation: the starting point.
- Error (lost) function.
- Find the (optimal) solution.

The goals and key ingredients in scientific computing

Approximate a mapping/function.

Key ingredients

- Representation: the starting point.
- Error (lost) function.
- Find the (optimal) solution.

The goals: efficiency, accuracy and stability.

The goals and key ingredients in scientific computing

Approximate a mapping/function.

Key ingredients

- Representation: the starting point.
- Error (lost) function.
- Find the (optimal) solution.

The goals: efficiency, accuracy and stability.

- These goals and ingredients are entwined!
- Needs a holistic study and balanced approach in practice!

Least square approximation

General form: given a target function $f(x), x \in D \subset \mathbb{R}^d$, and *a chosen* parametrized representation $h(x; \alpha)$

$$\min_{\alpha} I(\alpha) = \|h(\cdot; \alpha) - f(\cdot)\|^2_{L^2(D)}$$

Least square approximation

General form: given a target function $f(x), x \in D \subset \mathbb{R}^d$, and *a chosen* parametrized representation $h(x;\alpha)$

$$\min_{\alpha} I(\alpha) = \|h(\cdot;\alpha) - f(\cdot)\|^2_{L^2(D)}$$

Basic numerical questions of practical importance:

- the best accuracy one can achieve given a finite machine precision,
- the computation cost to achieve a given accuracy,
- stability with respect to perturbations.

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.
- Important numerical questions:

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.
- Important numerical questions: G!

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.
- Important numerical questions: G! G!

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.
- Important numerical questions: $G!$ $G!$ $G!$

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.
- Important numerical questions: $G!$ $G!$ $G!$
  - spectral property of G,
  - sparsity of G,
  - computation cost of G, $G^{\dagger}$.

Least square approximation

Linear representation: given a target function $f(x), x \in D \subset \mathbb{R}^d$, choose a set of basis functions $\psi_i(x)$, $h(x;\boldsymbol{\alpha}) = \sum_{i=1}^{n} a_i \psi_i(x)$, $\boldsymbol{\alpha} = (a_1, \ldots, a_n)^T$

$$\min_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = \| \sum_{i=1}^{n} a_i \psi_i(\cdot) - f(\cdot) \|^2_{L^2(D)}$$

$$\Rightarrow \boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}} I(\boldsymbol{\alpha}) = G^{\dagger}\mathbf{f},$$

G is Gram matrix, $G(i,j) = \langle \psi_i, \psi_j \rangle_D$, $\mathbf{f} = (\langle f, \psi_1 \rangle_D, \ldots, \langle f, \psi_n \rangle_D)^T$.

- Important mathematical questions: $V = \text{span}\{\psi_1, \ldots, \psi_n\}$, $\text{dist}(f, V)$.
- Important numerical questions: $G!$ $G!$ $G!$
  - spectral property of G,
  - sparsity of G,
  - computation cost of G, $G^{\dagger}$.
  - *choice of the basis is the key!*

Setup of neural networks (NN)

Two layer NN with `reLU` activation function $\sigma(t) = \max(0, t)$:

$$h(x) = \sum_{i=1}^{n} a_i \sigma(w_i \cdot x - b_i), \quad x \in \mathbb{R}^d.$$

We show

- ill-conditioning of the representation using global activation functions
- slow learning dynamics for high frequencies
- probabilistic characterization in parameter space

Setup of neural networks (NN)

Two layer NN with `reLU` activation function $\sigma(t) = \max(0, t)$:

$$h(x) = \sum_{i=1}^{n} a_i \sigma(w_i \cdot x - b_i), \quad x \in \mathbb{R}^d.$$

We show

- ill-conditioning of the representation using global activation functions
- slow learning dynamics for high frequencies
- probabilistic characterization in parameter space

which imply

- a two layer NN using global activation function is a ”low pass filter” in terms of representation and learning dynamics in practice

Setup of neural networks (NN)

Two layer NN with `reLU` activation function $\sigma(t) = \max(0, t)$:

$$h(x) = \sum_{i=1}^{n} a_i \sigma(w_i \cdot x - b_i), \quad x \in \mathbb{R}^d.$$

We show

- ill-conditioning of the representation using global activation functions
- slow learning dynamics for high frequencies
- probabilistic characterization in parameter space

which imply

- a two layer NN using global activation function is a ”low pass filter” in terms of representation and learning dynamics in practice
- numerical accuracy can be far from machine precision even if the network width goes to infinity

Setup of neural networks (NN)

Two layer NN with `reLU` activation function $\sigma(t) = \max(0, t)$:

$$h(x) = \sum_{i=1}^{n} a_i \sigma(w_i \cdot x - b_i), \quad x \in \mathbb{R}^d.$$

We show

- ill-conditioning of the representation using global activation functions
- slow learning dynamics for high frequencies
- probabilistic characterization in parameter space

which imply

- a two layer NN using global activation function is a ”low pass filter” in terms of representation and learning dynamics in practice
- numerical accuracy can be far from machine precision even if the network width goes to infinity
- what difference activation functions make

Setup of neural networks (NN)

Two layer NN with `reLU` activation function $\sigma(t) = \max(0, t)$:

$$h(x) = \sum_{i=1}^{n} a_i \sigma(w_i \cdot x - b_i), \quad x \in \mathbb{R}^d.$$

We show

- ill-conditioning of the representation using global activation functions
- slow learning dynamics for high frequencies
- probabilistic characterization in parameter space

which imply

- a two layer NN using global activation function is a ”low pass filter” in terms of representation and learning dynamics in practice
- numerical accuracy can be far from machine precision even if the network width goes to infinity
- what difference activation functions make
- why oscillatory functions are difficult to approximate

Two layer NN in 1D

$$h(x) = \sum_{i=1}^{n} a_i \sigma(x - b_i), \quad x, b_i \in D = (-1, 1), \; a_i \in \mathbb{R}$$

Mathematically, span$\{\sigma(x-b_i)\}$=span$\{P_1$ finite element basis$\}$.

Two layer NN in 1D

$$h(x) = \sum_{i=1}^{n} a_i \sigma(x - b_i), \quad x, b_i \in D = (-1, 1), \ a_i \in \mathbb{R}$$

Mathematically, span$\{\sigma(x-b_i)\}$=span$\{P_1$ finite element basis$\}$.
Numerically, the two sets of basis are very different!

Two layer NN in 1D

$$h(x) = \sum_{i=1}^{n} a_i \sigma(x - b_i), \quad x, b_i \in D = (-1, 1), \ a_i \in \mathbb{R}$$

Mathematically, span$\{\sigma(x-b_i)\}$=span$\{P_1$ finite element basis$\}$.
Numerically, the two sets of basis are very different!

- Finite element basis: local and almost orthogonal $\Rightarrow$ the Gram matrix is sparse and well conditioned (cond $= O(\frac{h_{max}}{h_{min}})$) $\Rightarrow$ capture all frequencies resolved by the mesh well.

Two layer NN in 1D

$$h(x) = \sum_{i=1}^{n} a_i \sigma(x - b_i), \quad x, b_i \in D = (-1, 1), \ a_i \in \mathbb{R}$$

Mathematically, span$\{\sigma(x-b_i)\}$=span$\{P_1$ finite element basis$\}$.
Numerically, the two sets of basis are very different!

- Finite element basis: local and almost orthogonal $\Rightarrow$ the Gram matrix is sparse and well conditioned (cond $= O(\frac{h_{max}}{h_{min}})$) $\Rightarrow$ capture all frequencies resolved by the mesh well.
- `ReLU` basis: global and can be highly correlated $\Rightarrow$ the Gram matrix is dense and has a fast spectral decay (ill-conditioned) $\Rightarrow$ only a certain number (depending on the machine precision or noise) of leading modes (low frequencies) can be captured stably in numerical computation $\Rightarrow$ low pass filter.

Spectral analysis for the Gram matrix of ReLU basis: 1D

The corresponding continuous kernel

$$\mathcal{G}(x,y)=\int_D \sigma(z-x)\sigma(z-y)dz=\frac{1}{12}|x-y|^3+\frac{1}{12}(2-x-y)\left(2(1-x)(1-y)-(x-y)^2\right)$$

The Gram matrix is $G_{ij}=(\mathcal{G}(b_i,b_j))_{1\le i,j\le n}$.

Lemma

The eigenvalues of $\mathcal{G}(x,y)$ in descending order are: $\mu_k = O(k^{-4})$. The corresponding eigenfunctions $\phi_k(x)$ satisfies

$$\phi_k^{(4)}(x)=\mu_k^{-1}\phi_k(x),\ x\in(-1,1),\quad \phi_k(1)=\phi_k^{(1)}(1)=\phi_k^{(2)}(-1)=\phi_k^{(3)}(-1)=0.$$

ϕ_k's are a combination of exponential functions and Fourier modes, which are asymptotically Fourier modes, from low to high frequencies.

Theorem

$\{b_i\}_{i=1}^n$ are i.i.d distributed with probability density function $\rho\in C^3[-1,1]$, $0<\underline{c}\le\rho(x)\le\bar{c}<\infty$. With probability $1-\frac{1}{n}$, the condition number $\lambda_1/\lambda_n\ge O(n^3(\log n)^{-1})$.

Spectral analysis for the Gram matrix of ReLU basis: $\mathbb{R}^d$

ReLU basis in $\mathbb{R}^d$: $\sigma(w \cdot x - b), w \in \mathbb{S}^{d-1}, b \in \mathbb{R}$. Define

$$\mathcal{G}((w,b),(w',b')) = \int_D \sigma(w \cdot x - b)\sigma(w' \cdot x - b')dx.$$

Use $\partial_b^2 \sigma(w \cdot x - b) = \Delta_x \sigma(w \cdot x - b) = \delta(w \cdot x - b)$, and Radon transform $\mathcal{R}$,

Lemma

The eigenfunction satisfies $\lambda_k \partial_b^2 \phi_k = \mathcal{R}\Delta^{-1}\mathcal{R}^* \phi_k$ in weak sense.
$(\lambda_k^{-1}, \mathcal{R}^*\phi_k)$ forms an eigen pair of the operator $\Delta^{-1}\mathcal{R}^*\mathcal{R}\Delta^{-1} = c_d(-\Delta)^{-\frac{d+3}{2}}$.

Theorem

Let λ_k be the eigenvalue of the kernel $\mathcal{G}$. There are constants $c_1, c_2 > 0$, depending on D and d, such that

$$c_1 k^{-(d+3)/d} \leq \lambda_k \leq c_2 k^{-(d+3)/d}.$$

- For σ^k, $\lambda_k = O(k^{-(d+k+2)/d})$.
- For analytic activation function, the spectral decays even faster.

Low pass filter

Assume $f(x) = \int_V \sigma(w \cdot x - b) h(w,b) dwdb \;\Rightarrow\; \Delta f(x) = \mathcal{R}^* h(w,b)$

$$h(w,b) = \sum_{k=1}^{\infty} \alpha_k \phi_k(w,x) \quad \Rightarrow \quad f(x) = \sum_{k=1}^{\infty} \alpha_k \Delta^{-1} \mathcal{R}^* \phi_k .$$

Key observations: $\mathcal{R}^* \phi_k$ are eigenfunctions of Δ.

Low pass filter

Assume $f(x) = \int_V \sigma(w \cdot x - b) h(w,b) dwdb \;\Rightarrow\; \Delta f(x) = \mathcal{R}^* h(w,b)$

$$h(w,b) = \sum_{k=1}^{\infty} \alpha_k \phi_k(w,x) \quad \Rightarrow \quad f(x) = \sum_{k=1}^{\infty} \alpha_k \Delta^{-1} \mathcal{R}^* \phi_k .$$

Key observations: $\mathcal{R}^* \phi_k$ are eigenfunctions of Δ.

- For machine precision ϵ, Moore-Penrose pseudo-inverse (MATLAB) maintains leading modes $k \leq m = \Theta(\epsilon^{-\frac{d}{2d+3}})$ of $h(w,b) \Rightarrow$ a low pass filter in the approximation of $f(x)$.

Low pass filter

Assume $f(x) = \int_V \sigma(w \cdot x - b) h(w,b) dwdb \;\Rightarrow\; \Delta f(x) = \mathcal{R}^* h(w,b)$

$$h(w,b) = \sum_{k=1}^{\infty} \alpha_k \phi_k(w,x) \quad \Rightarrow \quad f(x) = \sum_{k=1}^{\infty} \alpha_k \Delta^{-1} \mathcal{R}^* \phi_k .$$

Key observations: $\mathcal{R}^* \phi_k$ are eigenfunctions of Δ.

- For machine precision ϵ, Moore-Penrose pseudo-inverse (MATLAB) maintains leading modes $k \leq m = \Theta(\epsilon^{-\frac{d}{2d+3}})$ of $h(w,b) \Rightarrow$ a low pass filter in the approximation of $f(x)$.
- At most all eigenmodes of the Laplace operator up to frequency $\mathcal{O}(\epsilon^{-\frac{1}{2d+3}})$ can be captured accurately and stably in d dimensions.
  single precision $\epsilon = 2^{-23}$: $k_1 \simeq 24, k_2 \simeq 10, k_3 \simeq 6$.
  double precision $\epsilon = 2^{-52}$: $k_1 \simeq 1351, k_2 \simeq 172, k_3 \simeq 55, k_{10} = 5$.

Low pass filter

Assume $f(x) = \int_V \sigma(w \cdot x - b) h(w,b) dwdb \;\Rightarrow\; \Delta f(x) = \mathcal{R}^* h(w,b)$

$$h(w,b) = \sum_{k=1}^{\infty} \alpha_k \phi_k(w,x) \quad \Rightarrow \quad f(x) = \sum_{k=1}^{\infty} \alpha_k \Delta^{-1} \mathcal{R}^* \phi_k.$$

Key observations: $\mathcal{R}^* \phi_k$ are eigenfunctions of Δ.

- For machine precision ϵ, Moore-Penrose pseudo-inverse (MATLAB) maintains leading modes $k \leq m = \Theta(\epsilon^{-\frac{d}{2d+3}})$ of $h(w,b) \Rightarrow$ a low pass filter in the approximation of $f(x)$.
- At most all eigenmodes of the Laplace operator up to frequency $\mathcal{O}(\epsilon^{-\frac{1}{2d+3}})$ can be captured accurately and stably in d dimensions.
  single precision $\epsilon = 2^{-23}$: $k_1 \simeq 24, k_2 \simeq 10, k_3 \simeq 6$.
  double precision $\epsilon = 2^{-52}$: $k_1 \simeq 1351, k_2 \simeq 172, k_3 \simeq 55, k_{10} = 5$.
- numerical accuracy maybe far from machine precision for functions with significant high frequency components even if the network width goes to ∞.
- relative stable with respect to noise and over-parametrization.

Approximation error

- small network: $n \leq O(\epsilon^{-\frac{d}{2d+3}})$.
  The dominant error is due to discretization error:
  $h = O(n^{-\frac{1}{d}}) \leq O(\epsilon^{\frac{1}{2d+3}})$. The piecewise approximation error
  $\sim n^{-\frac{2}{d}} \|f\|_{H^2}$.
- large network: $n > O(\epsilon^{-\frac{d}{2d+3}})$.
  The dominant error is due to truncation error $\sim O(\epsilon^{\frac{p}{2d+3}} \|f\|_{H^p})$.

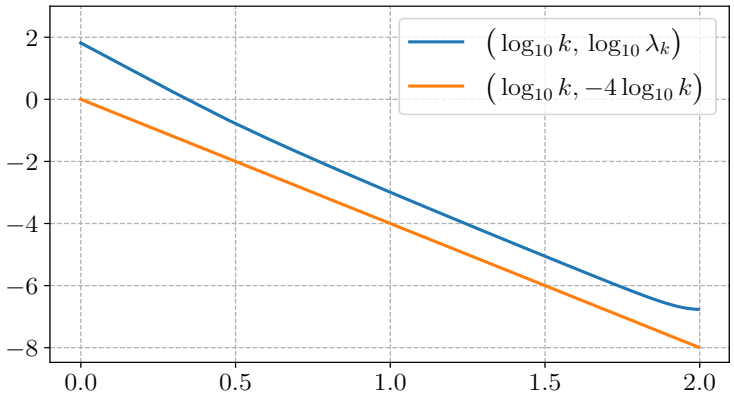
Approximation error

- small network: $n \leq O(\epsilon^{-\frac{d}{2d+3}})$.
  The dominant error is due to discretization error:
  $h = O(n^{-\frac{1}{d}}) \leq O(\epsilon^{\frac{1}{2d+3}})$. The piecewise approximation error
  $\sim n^{-\frac{2}{d}} \|f\|_{H^2}$.
- large network: $n > O(\epsilon^{-\frac{d}{2d+3}})$.
  The dominant error is due to truncation error $\sim O(\epsilon^{\frac{p}{2d+3}} \|f\|_{H^p})$.

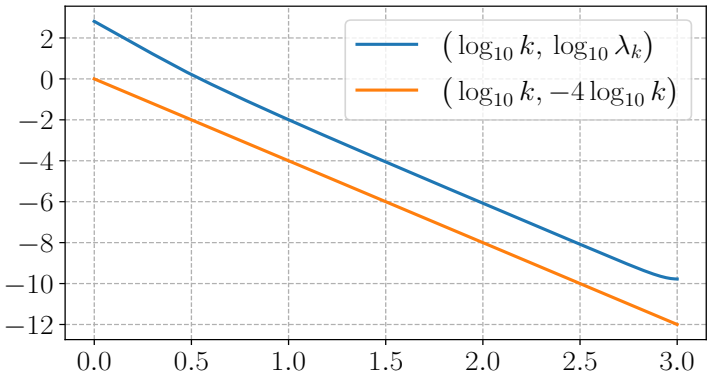
Remark

Using a smoother activation function, the spectral decay is even faster. It leads to larger truncation error for large networks but smaller discretization error in small network regime.

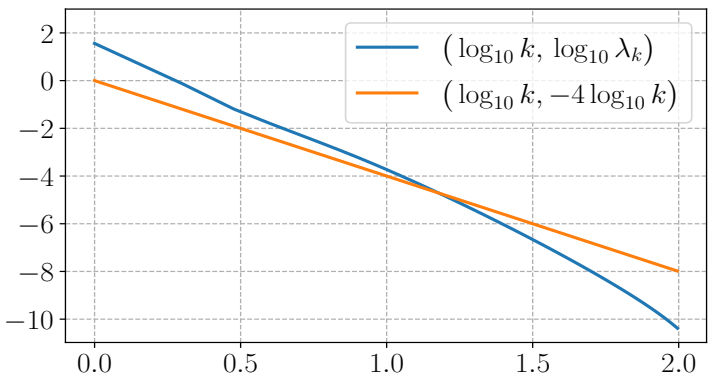
Numerical spectrum for Gram matrix (1D)



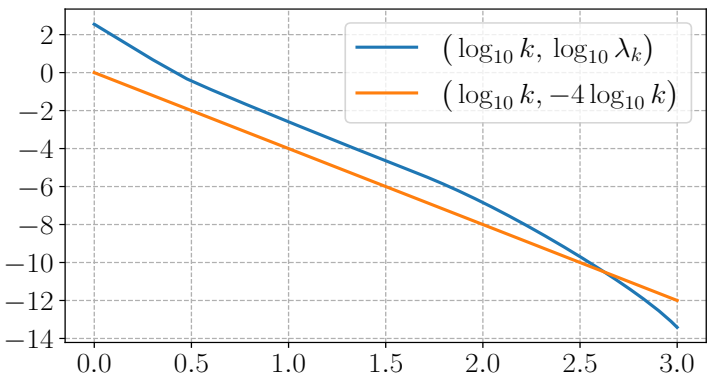
(a) n=100 (uniform bias)



(b) n=1000 (uniform bias)

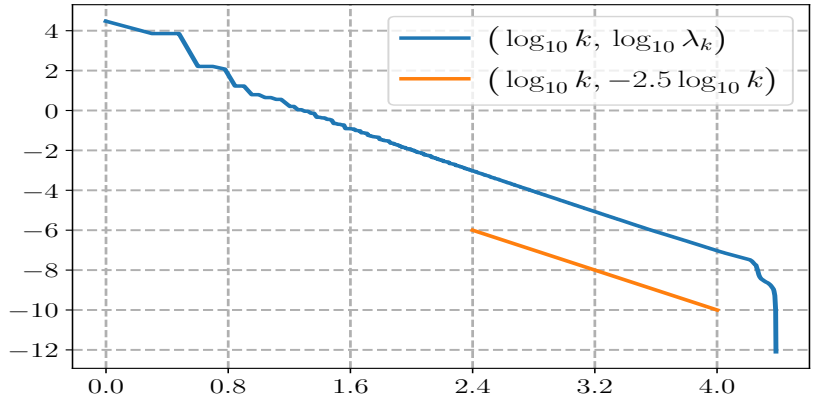


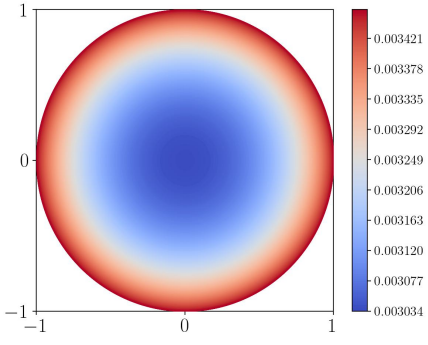
(a) n=100 (adaptive bias)



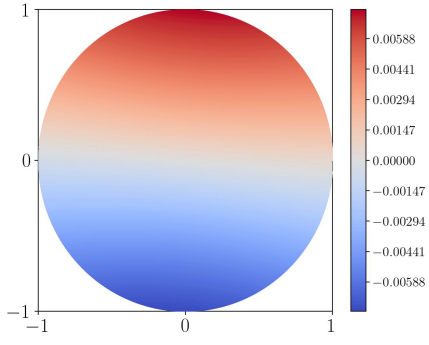
(b) n=1000 (adaptive bias)

Numerical spectrum for Gram matrix (2D)

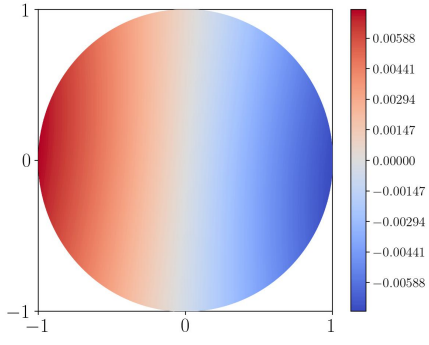




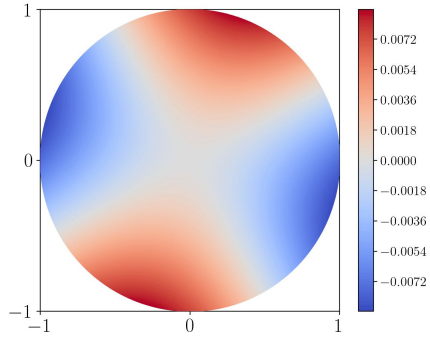
ϕ_1



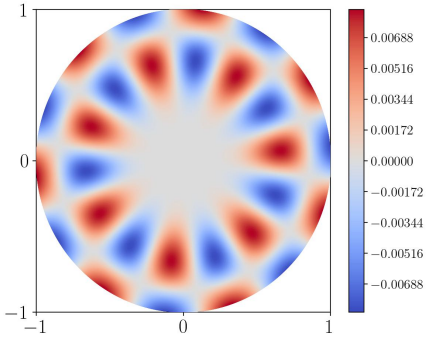
ϕ_2



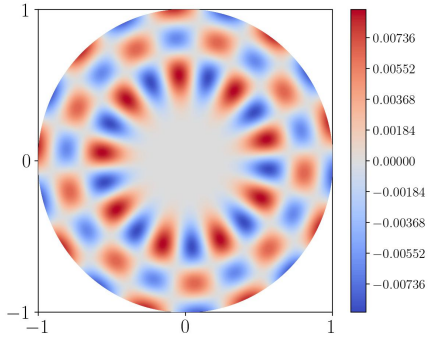
ϕ_3



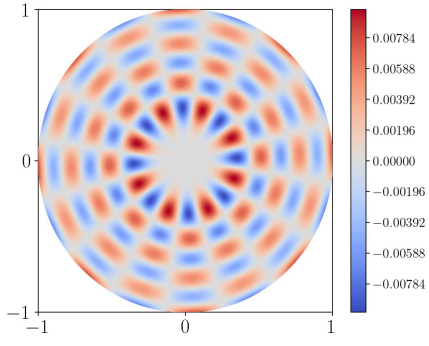
ϕ_4



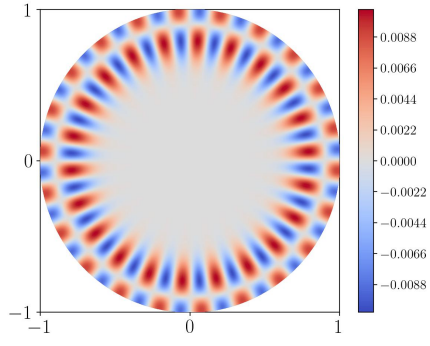
ϕ_{50}



ϕ_{100}



ϕ_{200}

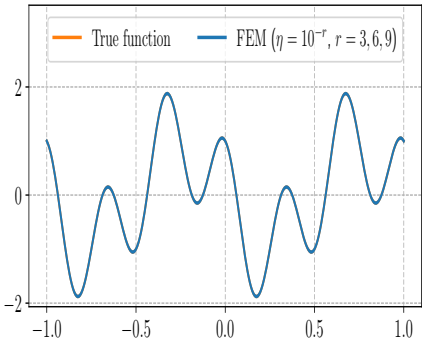


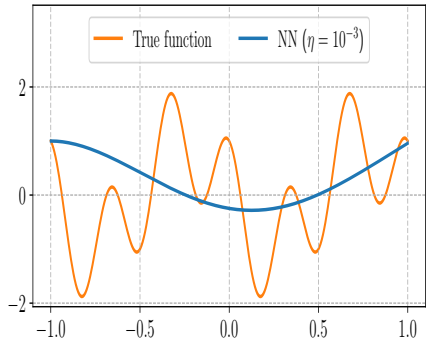
ϕ_{250}

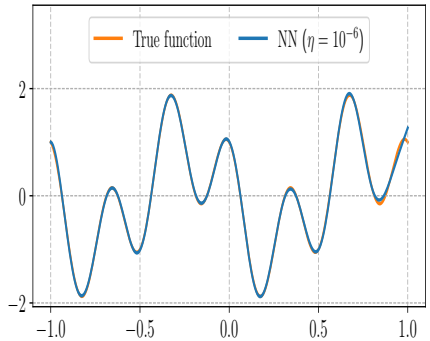
Numerical test

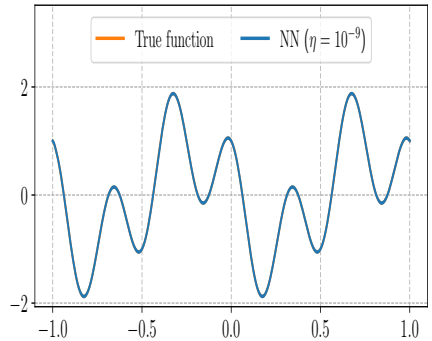
$f(x) = \cos(6\pi x) - \sin(2\pi x)$, $f_j(x) = f(jx)$. E-value threshold: $K = \max\{k : \frac{\lambda_k}{\lambda_1} \leq \eta\}$.

$f_1(x)$

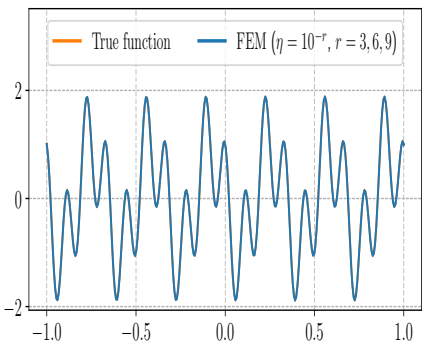


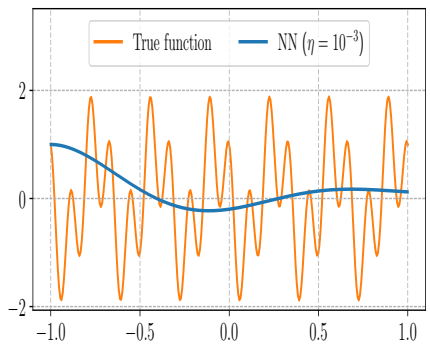


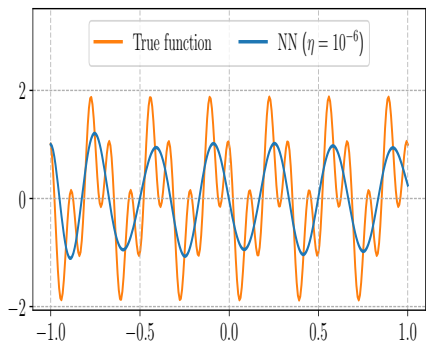


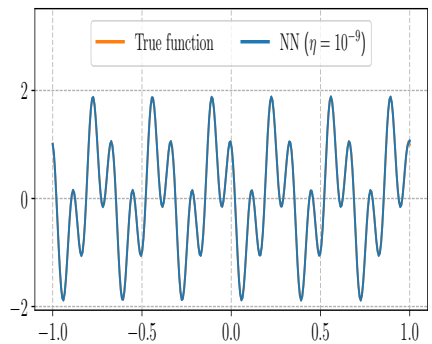


$f_3(x)$

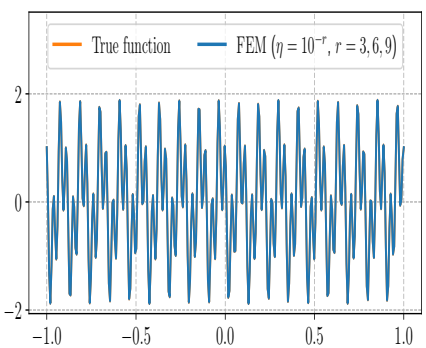


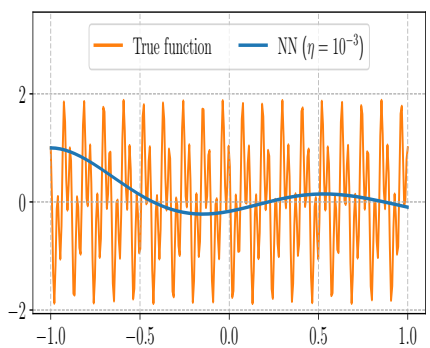


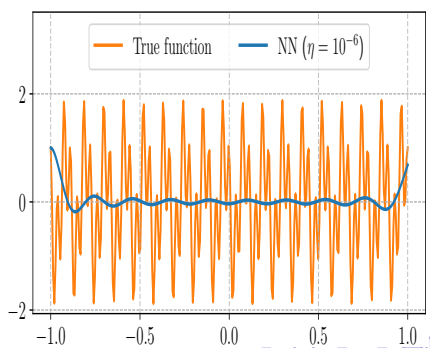


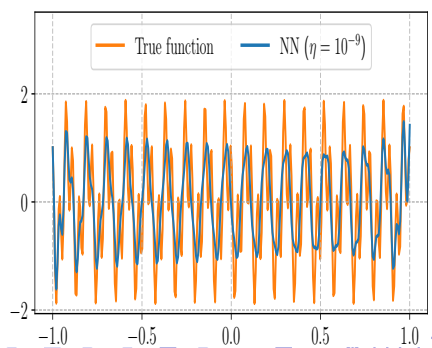


$f_9(x)$









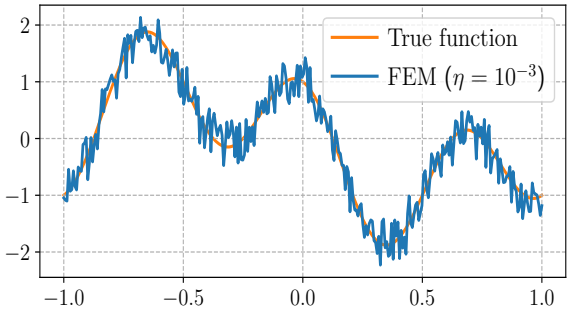
Numerical test

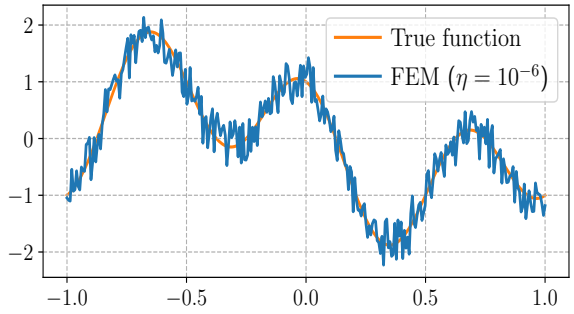
Table 1: Error comparison for approximating $f(x) = \arctan(25x)$ with sufficient samples.

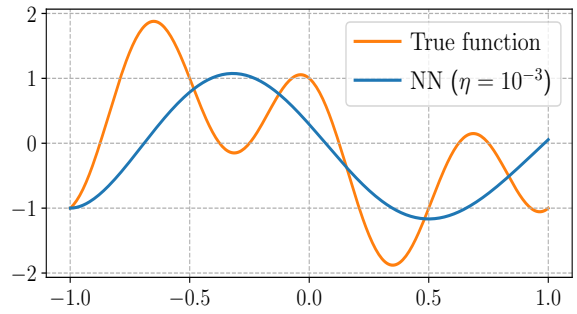
		float32				float64			
		$n = 100$		$n = 1000$		$n = 100$		$n = 1000$	
		MAX	MSE	MAX	MSE	MAX	MSE	MAX	MSE
NN	Uniform $\boldsymbol{b}$	6.09×10^{-2}	9.58×10^{-5}	7.19×10^{-2}	1.43×10^{-4}	1.37×10^{-2}	1.70×10^{-6}	1.05×10^{-4}	1.33×10^{-10}
FEM	Uniform $\boldsymbol{b}$	1.37×10^{-2}	1.70×10^{-6}	1.05×10^{-4}	1.33×10^{-10}	1.37×10^{-2}	1.70×10^{-6}	1.05×10^{-4}	1.33×10^{-10}
NN	Adaptive $\boldsymbol{b}$	6.83×10^{-2}	7.54×10^{-5}	1.89×10^{-2}	1.06×10^{-5}	3.93×10^{-3}	1.42×10^{-6}	4.74×10^{-5}	1.17×10^{-10}
FEM	Adaptive $\boldsymbol{b}$	2.92×10^{-3}	9.95×10^{-7}	3.79×10^{-5}	1.02×10^{-10}	2.92×10^{-3}	9.95×10^{-7}	3.77×10^{-5}	1.02×10^{-10}

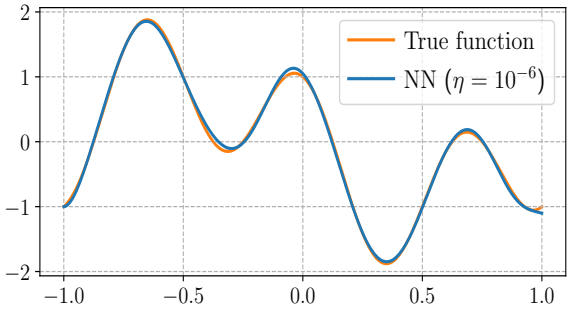
Stability with respect to noise and over-parametrization

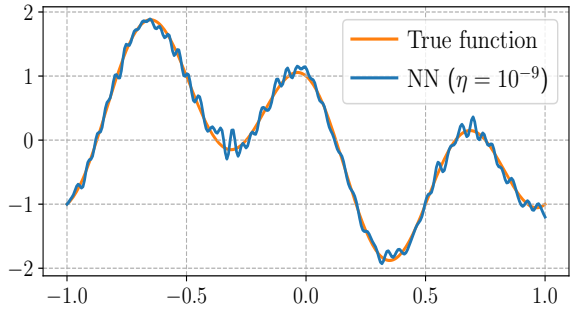
noise data

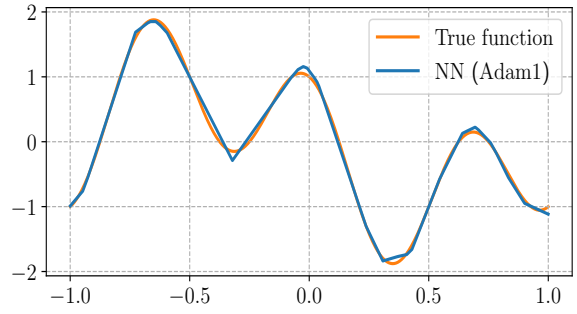




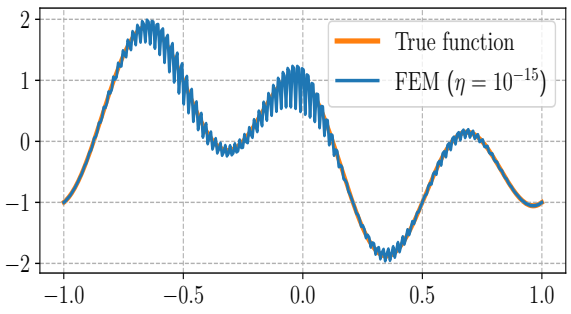


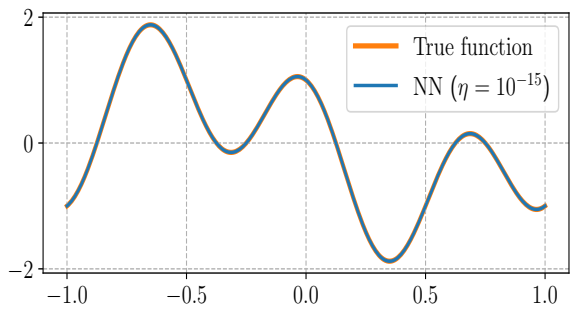


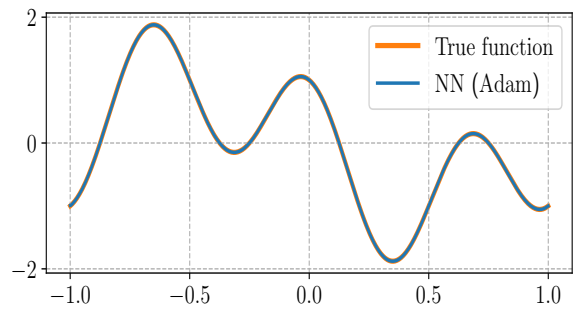




over-parametrization with 1000 samples and $n = 1500$



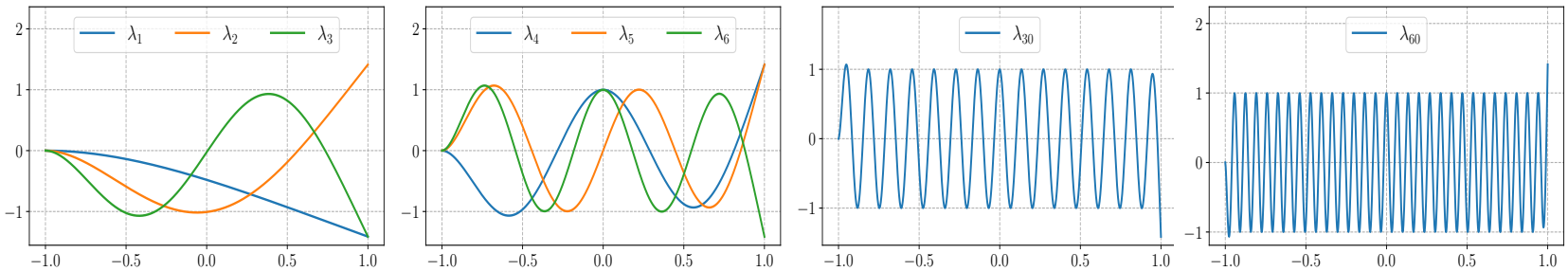




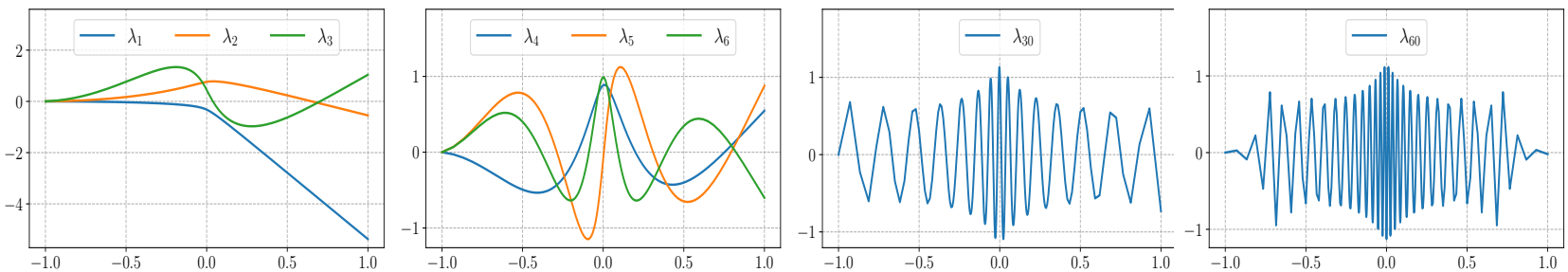
Adaptive vs uniform biases

Adaptive biases for $f(x) = \arctan(25x)$. Define $F(x) = \int_{-1}^{x} |f'(t)|dt \Big/ \int_{-1}^{1} |f'(t)|dt, \quad F(b_i) = (i-1)/(n-1).$

Eigenmodes of λ_k for $k = \{1,2,3\}, \{4,5,6\}, 30, 60$ with $n = 1000$.

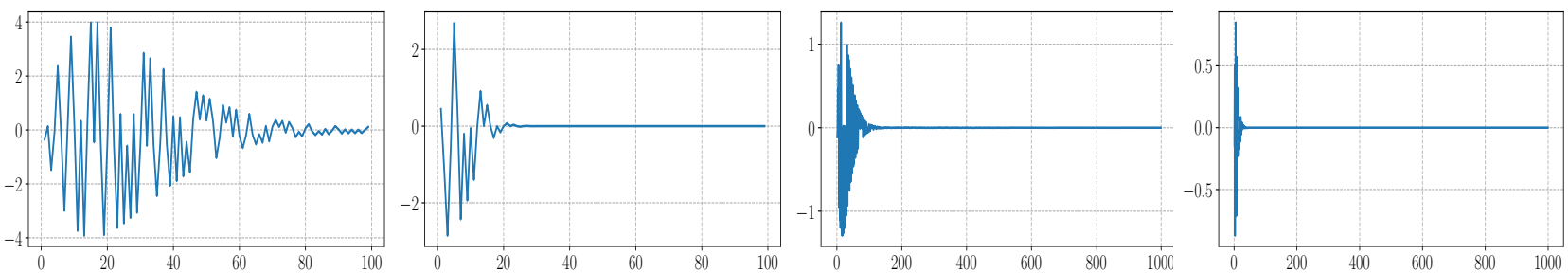


Uniformly distributed biases



adaptively distributed biases

Projection of f on the eigenmodes



uniform $n = 100$ | adaptive $n = 100$ | uniform $n = 1000$ | adaptive $n = 1000$

Learning dynamics based on gradient flow: linear

Training two layer ReLU neural networks: $h(x,t) = \sum_{i=1}^{n} a_i(t)\sigma(w_i \cdot x - b_i)$ following the gradient flow of $E(t) = \frac{1}{2}\|h(x,t) - f(x)\|^2_{L^2(D)}$

$$\frac{d\mathbf{a}(t)}{dt} = -G\mathbf{a}(t) + \mathbf{f}, \quad G_{ij} = < \sigma(w_i \cdot x - b_i), \sigma(w_j \cdot x - b_j) >_D, \mathbf{f}_i = < f(x), \sigma(w_i \cdot x - b_i) >$$

Let $(\lambda_k, \mathbf{g}_k)$ be the eigen pairs of G and define $\hat{a}_k(t) = \mathbf{a}^T(t)\mathbf{g}_k, \hat{f}_k = \mathbf{f}^T\mathbf{g}_k$.

$$\frac{d\hat{a}_k(t)}{dt} = -\lambda_k \hat{a}_k(t) + \hat{f}_k \quad \Rightarrow \quad \hat{a}_k(t) = (\hat{a}_k(0) - \frac{\hat{f}_k}{\lambda_k})e^{-\lambda_k t} + \frac{\hat{f}_k}{\lambda_k}.$$

Learning dynamics based on gradient flow: linear

Training two layer ReLU neural networks: $h(x,t) = \sum_{i=1}^{n} a_i(t)\sigma(w_i \cdot x - b_i)$ following the gradient flow of $E(t) = \frac{1}{2}\|h(x,t) - f(x)\|^2_{L^2(D)}$

$$\frac{d\mathbf{a}(t)}{dt} = -G\mathbf{a}(t) + \mathbf{f}, \quad G_{ij} = < \sigma(w_i \cdot x - b_i), \sigma(w_j \cdot x - b_j) >_D, \mathbf{f}_i = < f(x), \sigma(w_i \cdot x - b_i) >$$

Let $(\lambda_k, \mathbf{g}_k)$ be the eigen pairs of G and define $\hat{a}_k(t) = \mathbf{a}^T(t)\mathbf{g}_k, \hat{f}_k = \mathbf{f}^T\mathbf{g}_k$.

$$\frac{d\hat{a}_k(t)}{dt} = -\lambda_k \hat{a}_k(t) + \hat{f}_k \quad \Rightarrow \quad \hat{a}_k(t) = (\hat{a}_k(0) - \frac{\hat{f}_k}{\lambda_k})e^{-\lambda_k t} + \frac{\hat{f}_k}{\lambda_k}.$$

- It takes at least $t > \lambda_k^{-1}$ for the kth mode to converge.
- Noise will come in eventually in the long run.
- Stopping time plays the role of regularization.

Learning dynamics based on gradient flow: nonlinear

In general, training is the most challenging task for NN learning.

Learning dynamics based on gradient flow: nonlinear

In general, training is the most challenging task for NN learning.
Gradient descent for two layer neural networks
$h(x,t) = \sum_{i=1}^{n} a_i(t)\sigma(x - b_i(t))$

$$E(t) = \frac{1}{2}\|h(x,t) - f(x)\|_D^2, \quad D = (-1,1)$$

$$\frac{da_i}{dt} = -\int_D (h(x,t)-f(x))\sigma(x-b_i)dx \quad \frac{db_i}{dt} = a_i \int_D (h(x,t)-f(x))\sigma'(x-b_i)dx.$$

Learning dynamics based on gradient flow: nonlinear

In general, training is the most challenging task for NN learning.
Gradient descent for two layer neural networks
$h(x,t) = \sum_{i=1}^{n} a_i(t)\sigma(x - b_i(t))$

$$E(t) = \frac{1}{2}\|h(x,t) - f(x)\|_D^2, \quad D = (-1,1)$$

$$\frac{da_i}{dt} = -\int_D (h(x,t)-f(x))\sigma(x-b_i)dx \quad \frac{db_i}{dt} = a_i \int_D (h(x,t)-f(x))\sigma'(x-b_i)dx.$$

Basic questions:

- can the training process obtain the optimal a_i, b_i?

Learning dynamics based on gradient flow: nonlinear

In general, training is the most challenging task for NN learning.
Gradient descent for two layer neural networks
$h(x,t) = \sum_{i=1}^{n} a_i(t)\sigma(x - b_i(t))$

$$E(t) = \frac{1}{2}\|h(x,t) - f(x)\|_D^2, \quad D = (-1,1)$$

$$\frac{da_i}{dt} = -\int_D (h(x,t)-f(x))\sigma(x-b_i)dx \quad \frac{db_i}{dt} = a_i \int_D (h(x,t)-f(x))\sigma'(x-b_i)dx.$$

Basic questions:

- can the training process obtain the optimal a_i, b_i?
- what is the training dynamics and cost?

Learning dynamics based on gradient flow: nonlinear

Learning high frequencies is slow!

Theorem

It takes at least $O(m)$ time steps to get the initial error in (generalized) Fourier mode m reduced by half.

Learning dynamics based on gradient flow: nonlinear

Learning high frequencies is slow!

Theorem

It takes at least $O(m)$ time steps to get the initial error in (generalized) Fourier mode m reduced by half.

Main difficulties for the proof:

- fully nonlinear and discrete.
- dounded domain.

Remark

1. *The lower bound is not sharp.*
2. *With some mild conditions, the time step bound is $O(m^2)$.*
3. *With fixed biases, the time step bound is $O(m^4)$.*
4. *Smoother the activation function, the slower the training dynamics for high frequency components.*
5. *Experiments suggest Adam following a similar law initially.*

Rashomon set for two layer NN

Given a target function $f(x), x \in D = B_d(1)$. Denote $Q_{\mathcal{H}_n}$ to be the parameter domain for the two-layer ReLU neural network class

$$\mathcal{H}_n = \{h(x) | h(x) = \frac{1}{n} \sum_{j=1}^{n} a_j \sigma(w_j \cdot x - b_j), w_j \in \mathbb{S}^{d-1}, |a_j| \leq A, |b_j| \leq 1\}$$

The Rashomon set $\mathcal{R}_\epsilon(f) \subset Q_{\mathcal{H}_n}$

$$\mathcal{R}_\epsilon(f) := \{(w_j, a_j, b_j) \in Q_{\mathcal{H}_n}, s.t. \|h(\cdot; w_j, a_j, b_j) - f(\cdot)\|_{L^2(D)} \leq \epsilon \|f\|_{L^2(D)}\}$$

Normalize the measure on $Q_{\mathcal{H}_n}$, size of $\mathcal{R}_\epsilon(f)$ characterizes the likelihood that the loss is under certain threshold of relative error or how "easy" f can be approximated by $\mathcal{H}_n$.

Rashomon set for two layer NN

Theorem

Suppose $f \in C(D)$ such that there exists $g \in C_0^2(D)$ that $\Delta g = f$, then

$$\mathbb{P}(\mathcal{R}_\epsilon) \leq \exp\left(-\frac{n(1-\epsilon)^2\|f\|_{L^2(D)}^4}{2A^2\kappa^2}\right), \quad \kappa := \sup_{(w,b)} \int_{x\in D, w\cdot x = b\}} g(x) dH_{d-1}(x).$$

Remark

- *If f oscillates with frequency ν in all directions, then $\kappa \approx \nu^{-2}$ $\Rightarrow \mathbb{P}(\mathcal{R}_\epsilon) \sim \exp(-O(\nu^{-4}))$, which makes the approximation of oscillatory function difficult.*
- *Similar result holds for other bounded activation functions of the form $\sigma(w \cdot x - b)$.*

Key observations

$$h(x) = \frac{1}{n}\sum_{j=1}^{n} a_j \sigma(w_j \cdot x - b_j) \quad \Rightarrow \; \Delta h(x) = \frac{1}{n}\sum_{j=1}^{n} a_j \delta(w_j \cdot x - b_j)$$

$$\Rightarrow \; \langle h, f\rangle \overset{\Delta g = f}{=} \langle \Delta h, g\rangle = \frac{1}{n}\sum_{j=1}^{n} X_j, \quad X_j = a_j \int_{w_j \cdot x = b_j} g(x) dH_{d-1}(x)$$

X_j are i.i.d in $[-A\kappa, A\kappa]$, $E[X_j] = 0$, $\kappa := \sup_{(w,b)} \int_{\{x \in D, w\cdot x = b\}} g(x) dH_{d-1}(x)$

$$\mathbb{P}\Big[\|h-f\|_{L^2(D)} \le \epsilon \|f\|_{L^2(D)}\Big] \le \mathbb{P}\Big[\langle h, f\rangle \ge (1-\epsilon)\|f\|^2_{L^2(D)}\Big] \le \exp\left(-\frac{n(1-\epsilon)^2 \|f\|^4_{L^2(D)}}{2A^2\kappa^2}\right)$$

by Hoeffding's inequality

$$\mathbb{P}\left[\frac{1}{n}\sum_{j=1}^{n} X_j - E[X_j] \ge t\right] \le \exp\left(-\frac{nt^2}{2A^2\kappa^2}\right).$$

$\sigma(x)$ does not see oscillations well! $\langle \sigma, f\rangle = \int_{\{x \in D, w\cdot x = b\}} \Delta^{-1} f(x) dH_{d-1}(x)$

Take home messages for two layer networks

- Strongly correlated global activation functions lead to ill-conditioning of representation and slow learning dynamics for high frequencies.

Take home messages for two layer networks

- Strongly correlated global activation functions lead to ill-conditioning of representation and slow learning dynamics for high frequencies.
- Using global activation functions can capture smooth structures with sparse sampling $\Rightarrow$ global in physical domain while local in frequency domain.

Take home messages for two layer networks

- Strongly correlated global activation functions lead to ill-conditioning of representation and slow learning dynamics for high frequencies.
- Using global activation functions can capture smooth structures with sparse sampling $\Rightarrow$ global in physical domain while local in frequency domain.
- Using local activation functions can capture local and fine features with large degrees of freedom and dense sampling $\Rightarrow$ local in physical domain while global in frequency domain.

Take home messages for two layer networks

- Strongly correlated global activation functions lead to ill-conditioning of representation and slow learning dynamics for high frequencies.
- Using global activation functions can capture smooth structures with sparse sampling $\Rightarrow$ global in physical domain while local in frequency domain.
- Using local activation functions can capture local and fine features with large degrees of freedom and dense sampling $\Rightarrow$ local in physical domain while global in frequency domain.
- Approximation in problem specific transformed domain can help.

Take home messages for two layer networks

- Strongly correlated global activation functions lead to ill-conditioning of representation and slow learning dynamics for high frequencies.
- Using global activation functions can capture smooth structures with sparse sampling $\Rightarrow$ global in physical domain while local in frequency domain.
- Using local activation functions can capture local and fine features with large degrees of freedom and dense sampling $\Rightarrow$ local in physical domain while global in frequency domain.
- Approximation in problem specific transformed domain can help.

Remark.

- Training highly adpative w's and b's is not effective.
- Training a's with fixed random w, b to approximate smooth functions is effective.

Take home messages for two layer networks

- Strongly correlated global activation functions lead to ill-conditioning of representation and slow learning dynamics for high frequencies.
- Using global activation functions can capture smooth structures with sparse sampling $\Rightarrow$ global in physical domain while local in frequency domain.
- Using local activation functions can capture local and fine features with large degrees of freedom and dense sampling $\Rightarrow$ local in physical domain while global in frequency domain.
- Approximation in problem specific transformed domain can help.

Remark.

- Training highly adpative w's and b's is not effective.
- Training a's with fixed random w, b to approximate smooth functions is effective.

Reference:
Why Shallow Networks Struggle with Approximating and Learning High Frequency,

S. Zhang, H. Zhao, Y. Zhong and H. Zhou. arXiv:2306.17301, 2023.

Structured Decomposition and Approximation by Multi-layer Networks

Using large deep over-parametrized black-box type of networks causes difficulties in optimization, stability, and interpretation. The key idea is to introduce structure and balance in the network based on

- representation of a complicated function by smooth decomposition/transformation,
- multilayer network based on composition of structured and balanced shallow networks,

$\Rightarrow$ small network
$\Rightarrow$ better accuracy and efficiency in terms of degrees of freedom and more effective training dynamics.

Examples

Example 1: $f(x) = e^{-(10x)^2}, \quad x \in [-1, 1]$.

$$f_1(x) = \begin{cases} \frac{x-8}{9} & x \in [-1, -0.1] \\ 9x & x \in [-0.1, 0.1] \\ \frac{x+8}{9} & x \in [0.1, 1] \end{cases}$$

$$f_2(x) = \begin{cases} e^{-[10(9x+8)]^2} & x \in [-1. - 0.9] \\ e^{-(\frac{10x}{9})^2} & x \in [-0.9, 0.9] \\ e^{-[10(9x-8)]^2} & x \in [0.9, 1] \end{cases}$$

$$f(x) = f_2 \circ f_1(x) \quad x \in [-1, 1]$$

Examples

Example 1: $f(x) = e^{-(10x)^2}, \quad x \in [-1, 1]$.

$$f_1(x) = \begin{cases} \frac{x-8}{9} & x \in [-1, -0.1] \\ 9x & x \in [-0.1, 0.1] \\ \frac{x+8}{9} & x \in [0.1, 1] \end{cases}$$

$$f_2(x) = \begin{cases} e^{-[10(9x+8)]^2} & x \in [-1. - 0.9] \\ e^{-(\frac{10x}{9})^2} & x \in [-0.9, 0.9] \\ e^{-[10(9x-8)]^2} & x \in [0.9, 1] \end{cases}$$

$$f(x) = f_2 \circ f_1(x) \quad x \in [-1, 1]$$

Example2: $f(x) = \Big(\cos(2n\pi x) - 1\Big)/2, \quad x \in [-1, 1]$

$$f_1(x) = \begin{cases} \Big(\cos(2n\pi x) - 1\Big)/2 & \text{for } x \in [-1, 0), \\ x & \text{for } x \in [0, 1]. \end{cases}$$

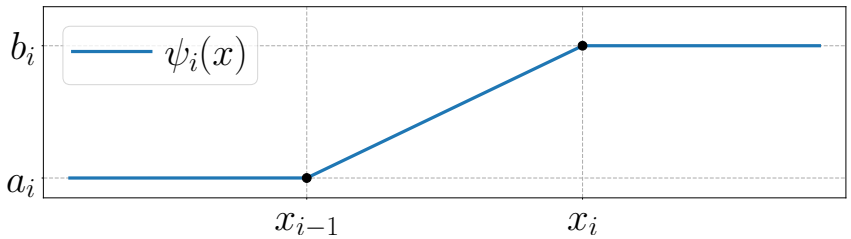
$$f_2(x) = \begin{cases} x & \text{for } x \in [-1, 0), \\ \Big(\cos(2n\pi x) - 1\Big)/2 & \text{for } x \in [0, 1]. \end{cases}$$

$$f(x) = f_2 \circ f_1(x) \quad x \in [-1, 1]$$

Mathematical formulation

1. divide: $-1 = x_0 < x_1 < \cdots < x_n = 1$, define

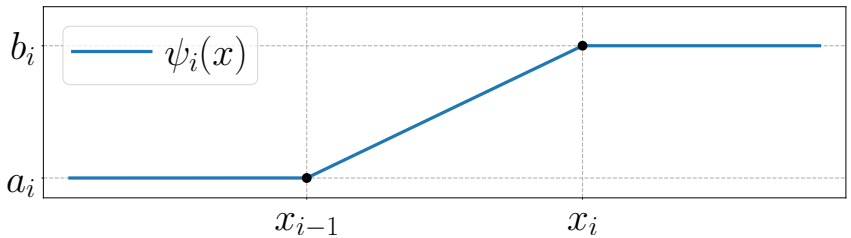
$$\psi_i(x) = s_i \cdot \mathrm{ReLU}\left(x - x_{i-1}\right) - s_i \cdot \mathrm{ReLU}\left(x - x_i\right) + a_i, \quad s_i = \frac{b_i - a_i}{x_i - x_{i-1}}$$



Mathematical formulation

1. divide: $-1 = x_0 < x_1 < \cdots < x_n = 1$, define

$$\psi_i(x) = s_i \cdot \mathrm{ReLU}\left(x - x_{i-1}\right) - s_i \cdot \mathrm{ReLU}\left(x - x_i\right) + a_i, \quad s_i = \frac{b_i - a_i}{x_i - x_{i-1}}$$



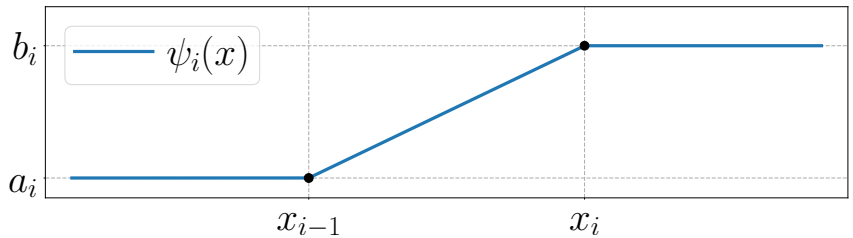
2. and conquer (by scaling): $\mathcal{L}_i : [a_i, b_i] \to [x_{i-1}, x_i]$ a linear function. $f_i = f \circ \mathcal{L}_i : [a_i, b_i] \to \mathcal{R}$ is a smoother function which can be approximated by one hidden layer network more easily.

$$f(x) = \sum_{i=1}^{n} f_i \circ \psi_i(x) - \underbrace{\sum_{i=1}^{n-1} f(x_i)}_{\text{constant}}$$

Mathematical formulation

1. divide: $-1 = x_0 < x_1 < \cdots < x_n = 1$, define

$$\psi_i(x) = s_i \cdot \mathrm{ReLU}\left(x - x_{i-1}\right) - s_i \cdot \mathrm{ReLU}\left(x - x_i\right) + a_i, \quad s_i = \frac{b_i - a_i}{x_i - x_{i-1}}$$



2. and conquer (by scaling): $\mathcal{L}_i : [a_i, b_i] \to [x_{i-1}, x_i]$ a linear function. $f_i = f \circ \mathcal{L}_i : [a_i, b_i] \to \mathcal{R}$ is a smoother function which can be approximated by one hidden layer network more easily.

$$f(x) = \sum_{i=1}^{n} f_i \circ \psi_i(x) - \underbrace{\sum_{i=1}^{n-1} f(x_i)}_{\text{constant}}$$

Dividing + scaling + approximation of smooth function can be achieved by composition of single layer networks!

Structured and balanced multi-layer networks

A single hidden layer network is viewed as a mapping: $\mathbb{R}^d \to \mathbb{R}$,

$$h(\mathbf{x}) = \sum_{j=1}^{n} a_j \sigma(\mathbf{w}_j \cdot \mathbf{x} - b_j) + c, \quad \mathbf{w}_j, \mathbf{x} \in \mathbb{R}^d, a_j, b_j, c \in \mathbb{R}.$$

Structured and balanced multi-layer networks

A single hidden layer network is viewed as a mapping: $\mathbb{R}^d \to \mathbb{R}$,

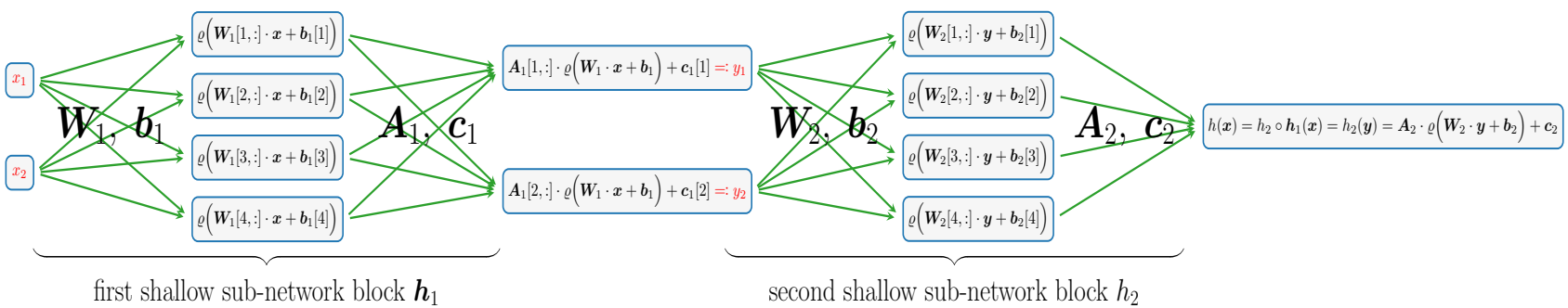
$$h(\mathbf{x}) = \sum_{j=1}^{n} a_j \sigma(\mathbf{w}_j \cdot \mathbf{x} - b_j) + c, \quad \mathbf{w}_j, \mathbf{x} \in \mathbb{R}^d, a_j, b_j, c \in \mathbb{R}.$$

Each layer l is composed of (rank) d_l single layer networks,

$$h_i^l(\mathbf{x}) = \sum_{j=1}^{n_l} a_{i,j}^l \sigma(\mathbf{w}_j^l \cdot \mathbf{x} - b_j^l) + c_i^l, \quad i = 1, 2, \ldots, d_l, \ \mathbf{x} \in \mathbb{R}^{d_{l-1}},$$

In matrix form, $\mathbf{h}^l(\mathbf{x}) = [h_1^l(\mathbf{x}), \ldots, h_{d_l}^l(\mathbf{x})]^T : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$.

$$\mathbf{h}^l = \mathbf{A}^l \sigma(\mathbf{W}^l \mathbf{x} + \mathbf{B}^l) + \mathbf{c}^l, \quad \mathbf{A}^l \in \mathbb{R}^{d_l \times n_l}, \mathbf{W}^l \in \mathbb{R}^{n_l \times d_{l-1}}, \mathbf{B}^l \in \mathbb{R}^{n_l}, \mathbf{c}^l \in \mathbb{R}^{d_l}.$$



$$x \xrightarrow[h_1]{W_1,\ b_1,\quad A_1,\ c_1} A_1\cdot\varrho\left(W_1\cdot x + b_1\right) + c_1 =: h_1(x) = y \xrightarrow[h_2]{W_2,\ b_2,\quad A_2,\ c_2} A_2\cdot\varrho\left(W_2\cdot y + b_2\right) + c_2 =: h_2(y)$$

Structured and balanced multi-layer networks

Multi-layer structure: $\mathbf{h} = \mathbf{h}_L \circ \cdots \circ \mathbf{h}_2 \circ \mathbf{h}_1$

$$\mathbf{h}^l = \mathbf{A}^l \sigma(\mathbf{W}^l \mathbf{x} + \mathbf{B}^l) + \mathbf{c}^l, \quad \mathbf{A}^l \in \mathbb{R}^{d_l \times n_l}, \mathbf{W}^l \in \mathbb{R}^{n_l \times d_{l-1}}, \mathbf{B}^l \in \mathbb{R}^{n_l}, \mathbf{c}^l \in \mathbb{R}^{d_l}.$$

Key features:

- The multi-layer network is composed of single layers through channels (horizontally) and depth (vertically).
- Learning/optimizing $\mathbf{A}^l$ for each single layer network with random (fixed) $\mathbf{W}^l, \mathbf{B}^l, \mathbf{c}^l$ can approximate a smooth function effectively.
- Decomposition through channels and composition through depth is effective in dealing with complicated features,
- Using ADAM optimization generates interesting learning dynamics.

Remark

Each layer of a standard fully connected network,

$$\mathbf{h}^l(\mathbf{x}) = \sigma(\mathbf{W}^l \mathbf{x} + \mathbf{b}^l) + \mathbf{c}^l.$$

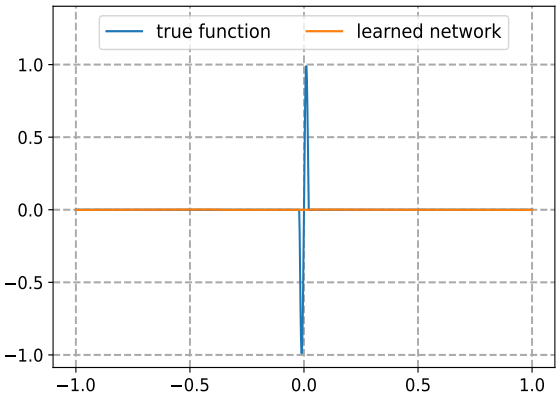
$\mathbf{W}^l$ *is of dimension* $n_l \times n_{l-1} \gg n_l \times d_{l-1}$.

Experiments: learning adaptivity

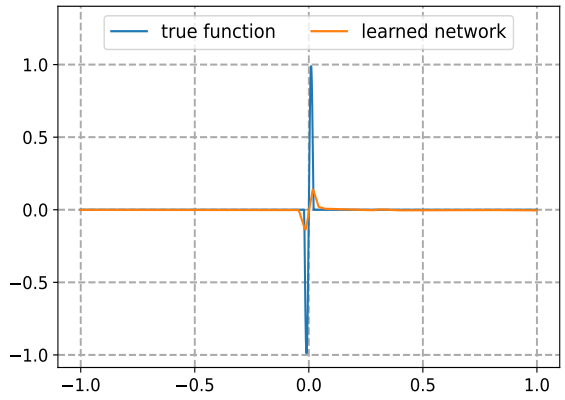
$f(x) = 1_{\{|x|<0.02\}} \cdot \sin(50\pi x)$ with 1000 uniformly sampled data.

Network: (width, rank, depth)=(10, 3, 5)

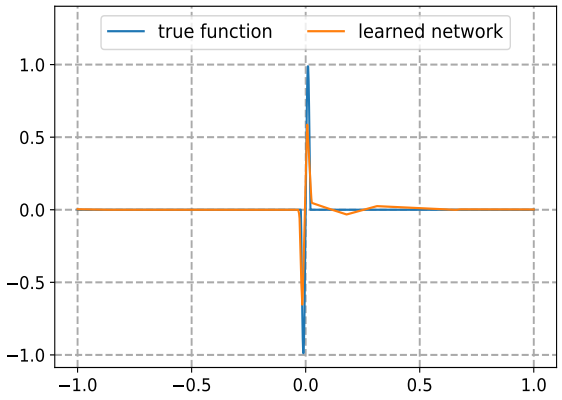
epoch 3000



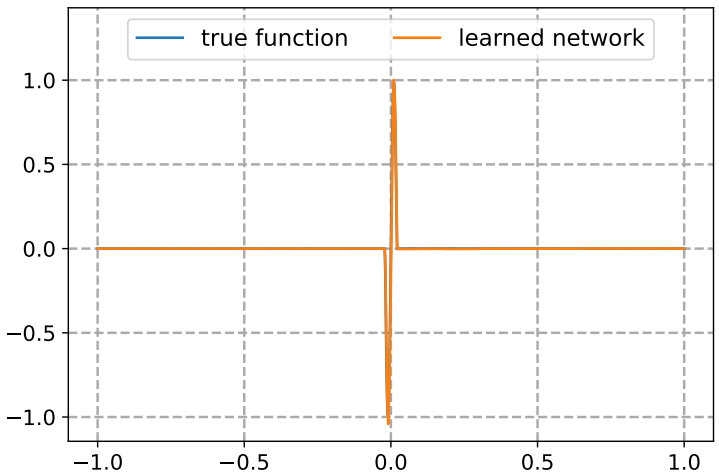
epoch 3500



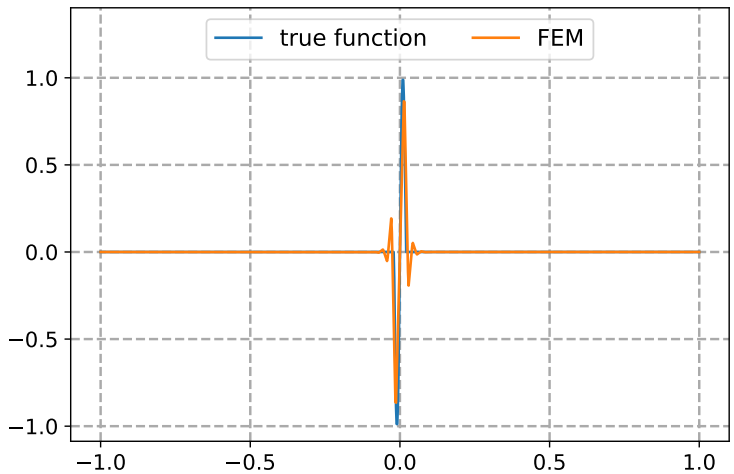
epoch 4000



epoch 10000



FEM with the same number of d.o.f

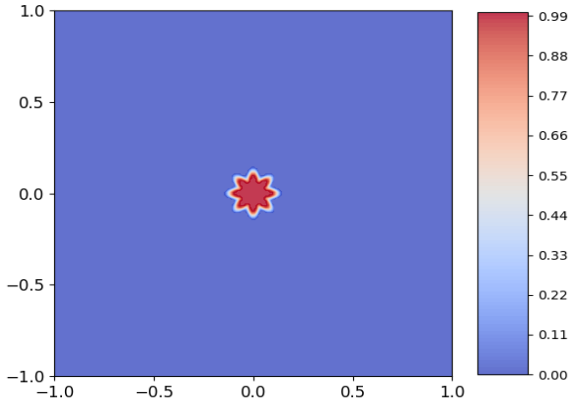


Experiments: learning adaptivity

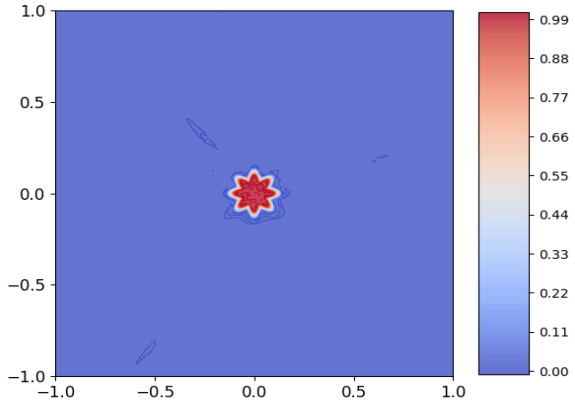
Target function: $f(r,\theta) = \begin{cases} 0 & \text{if } 25\rho - 25r + 0.5 \le 0, \\ 1 & \text{if } 25\rho - 25r + 0.5 \ge 1, \\ 5\rho - 5r + 0.5 & \text{otherwise,} \end{cases} \quad \rho = 0.1 + 0.02\cos(\pi\theta^2)$ with 400×400 samples.

Network: (width, rank, depth)=(100, 10, 6)

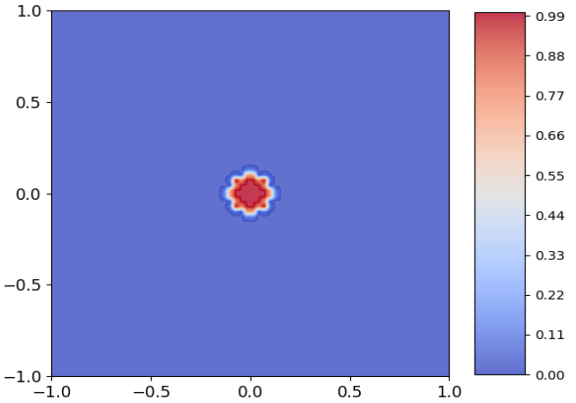
original function



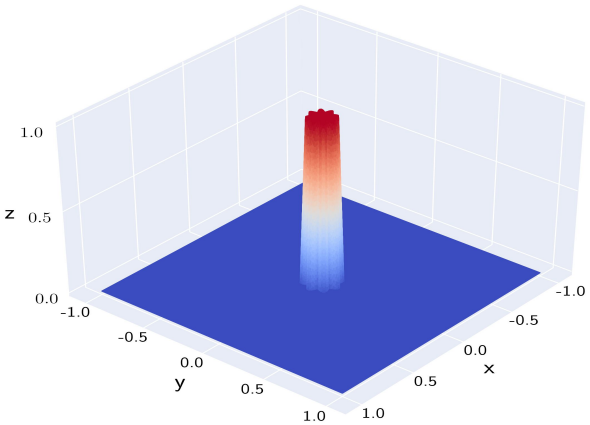
NN approximation



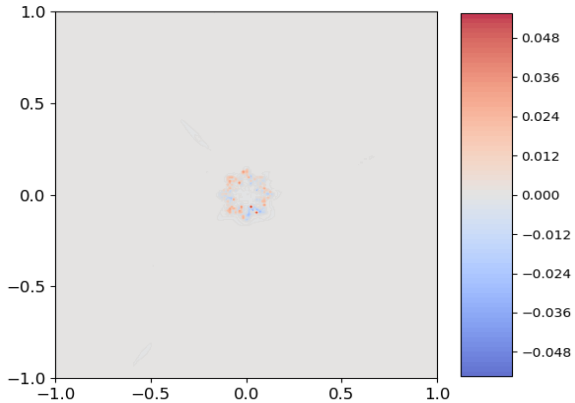
FEM with the same d.o.f.



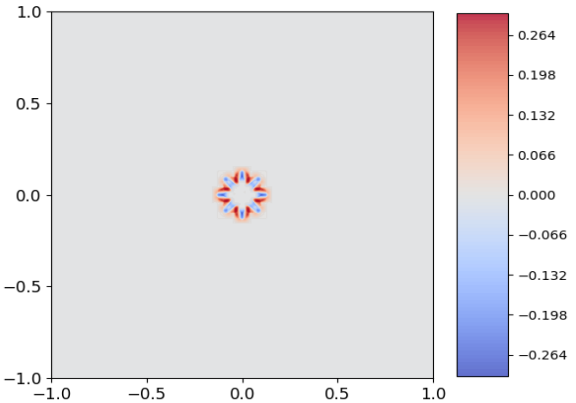
original function



NN error



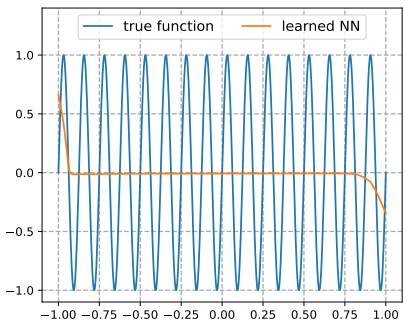
FEM error



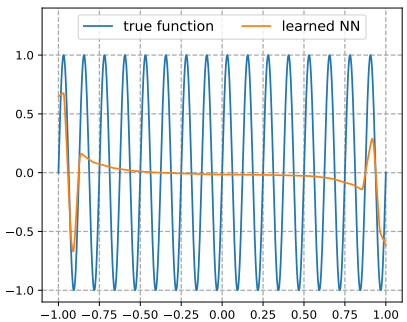
Experiments

target function: $f(x) = \sin(16\pi x)$
NN: (width, depth, rank)=(300, 4, 5)

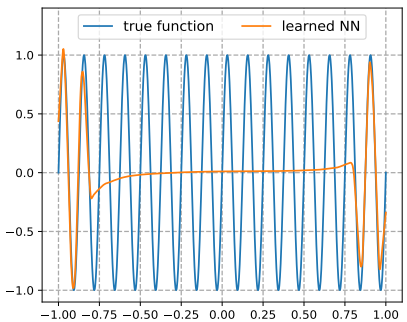
epoch 1



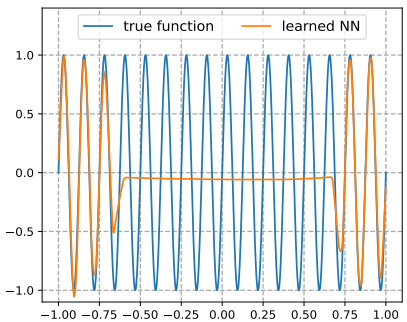
epoch 2



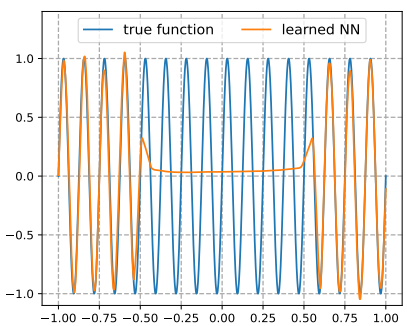
epoch 3



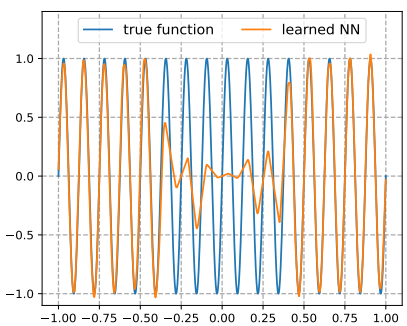
epoch 4



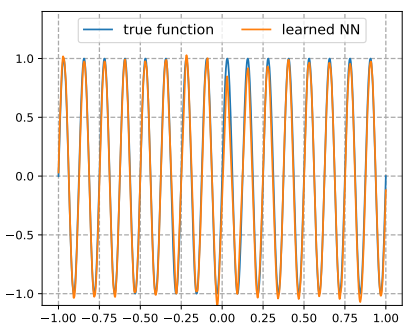
epoch 10



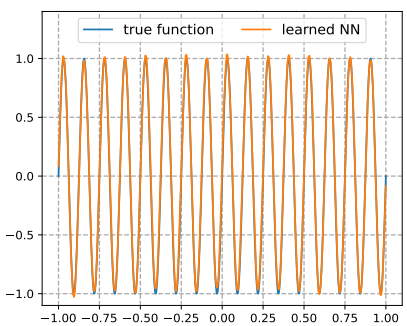
epoch 20



epoch 25



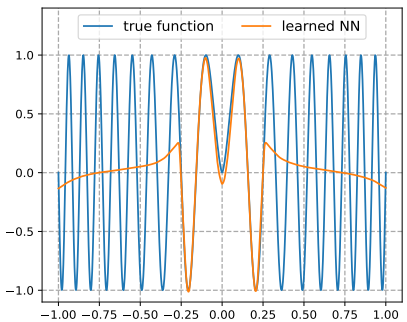
epoch 30



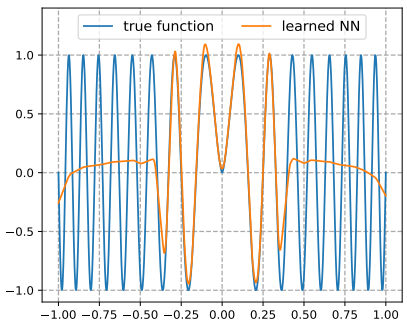
Experiments

target function: $f(x) = \sin(16\pi|x|^{\frac{3}{2}})$
NN: (width, depth, rank)=(300, 4, 5)

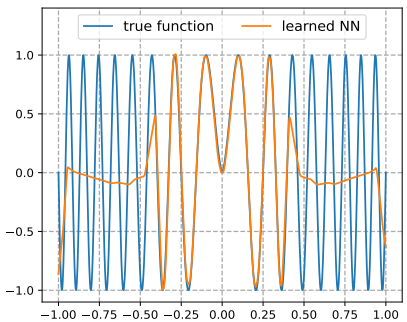
epoch 1



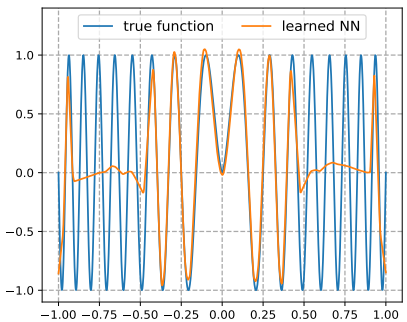
epoch 2



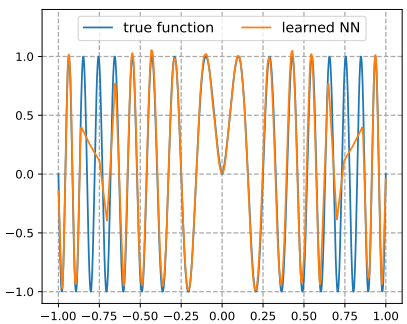
epoch 3



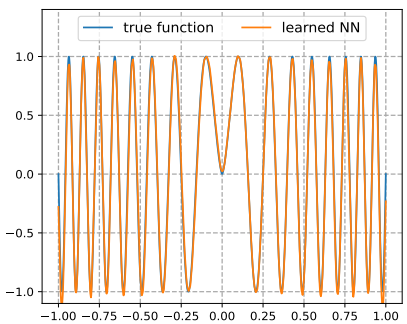
epoch 4



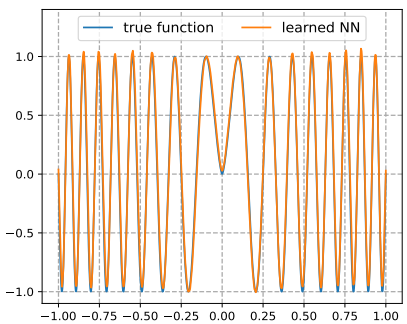
epoch 10



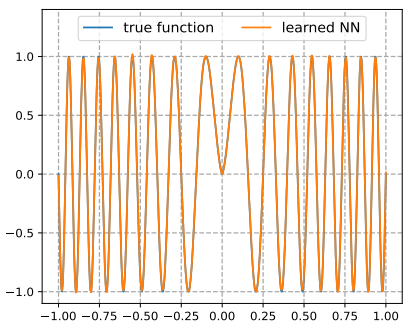
epoch 20



epoch 25



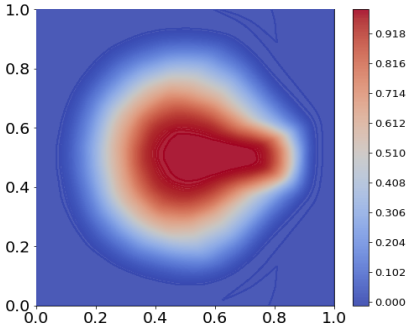
epoch 30



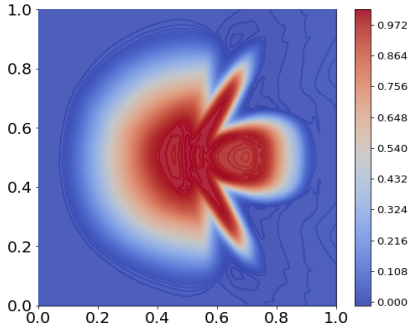
Experiments

NN: (width, depth, rank)=(500,5,3)

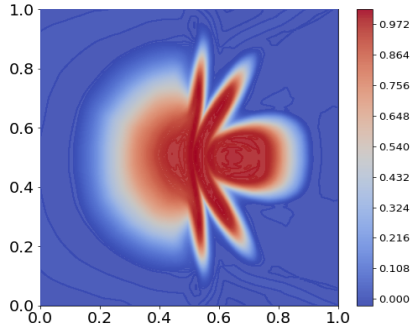
epoch 1



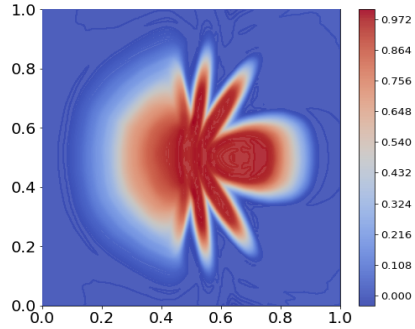
epoch 2



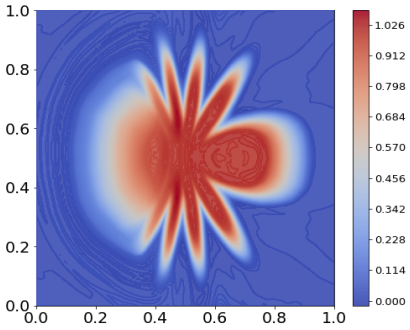
epoch 3



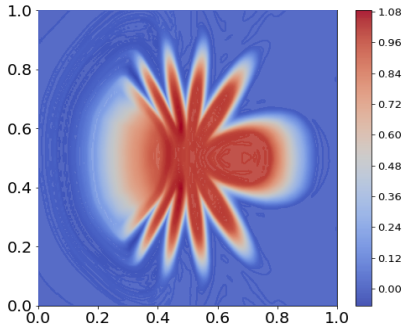
epoch 4



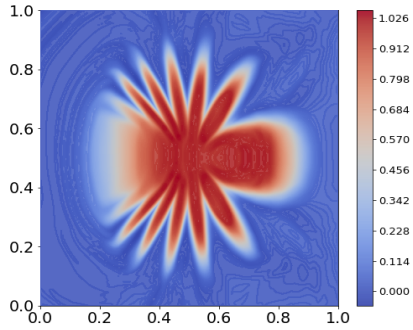
epoch 5



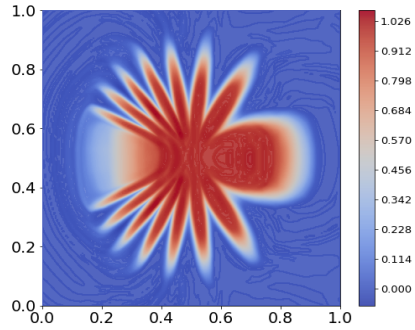
epoch 6



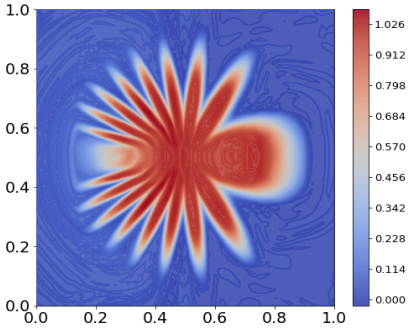
epoch 7



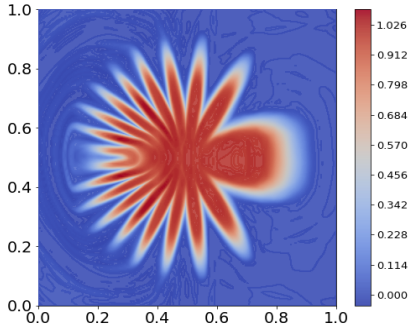
epoch 8



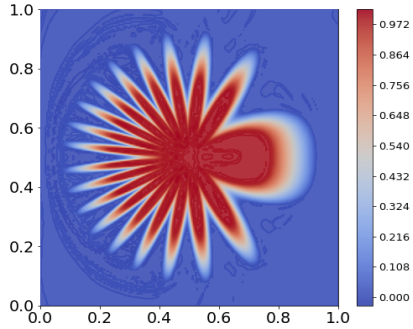
epoch 9



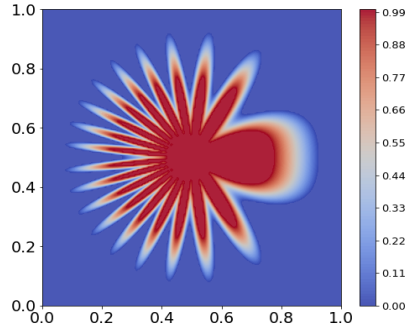
epoch 10



epoch 30



true function



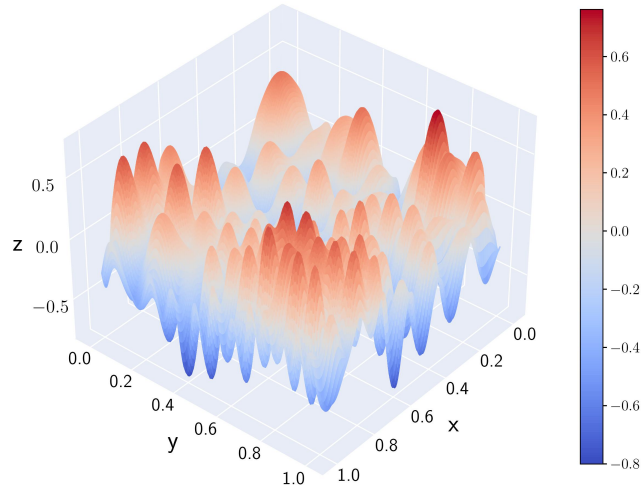
Experiments

$$f(x_1, x_2) = \sum_{i=1}^{2} \sum_{j=1}^{2} a_{ij} \sin(b_i x_i + c_{i,j} x_i x_j) \cos(b_j x_j + d_{i,j} x_i^2)$$

$$(a_{i,j}) = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix} \quad (b_i) = \begin{bmatrix} 2.4\pi \\ 4.8\pi \end{bmatrix},$$

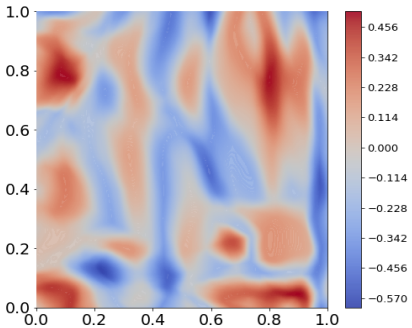
$$(c_{i,j}) = \begin{bmatrix} 2.4\pi & 4.8\pi \\ 9.6\pi & 4.8\pi \end{bmatrix} \quad (d_{i,j}) = \begin{bmatrix} 4.8\pi & 7.2\pi \\ 9.6\pi & 7.2\pi \end{bmatrix}.$$

NN: (width, depth, rank)=(500,5,5)

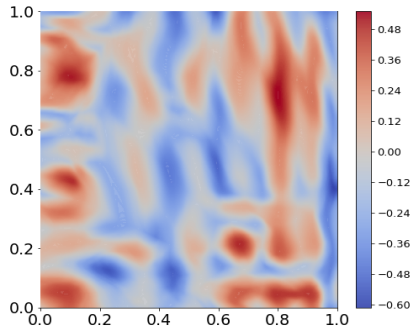


]

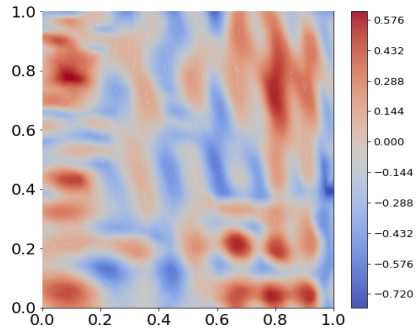
epoch 2



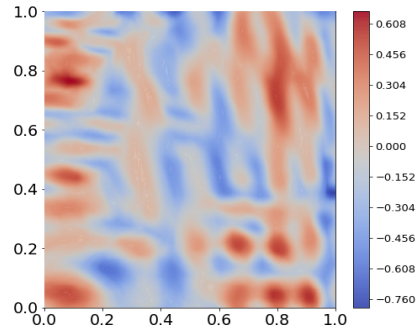
epoch 4



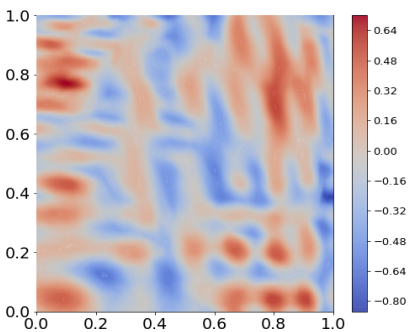
epoch 6



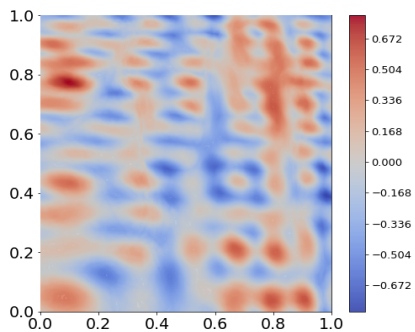
epoch 8



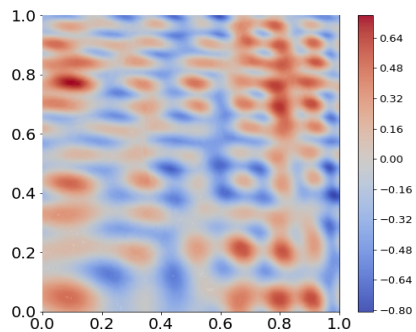
epoch 10



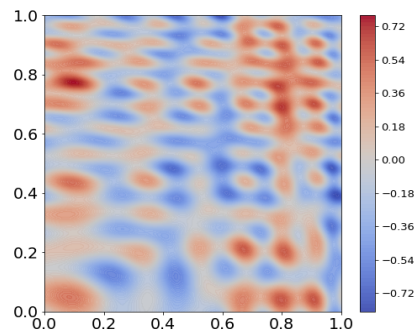
epoch 30



epoch 60



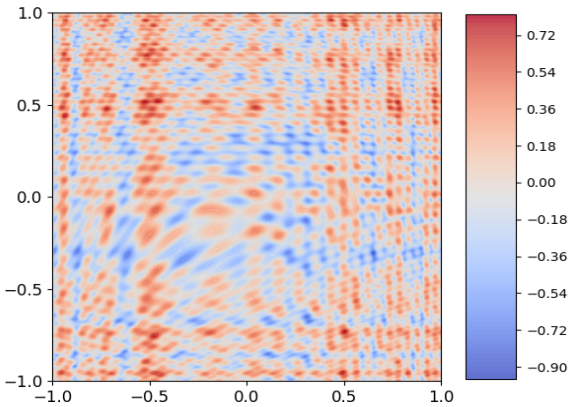
true function



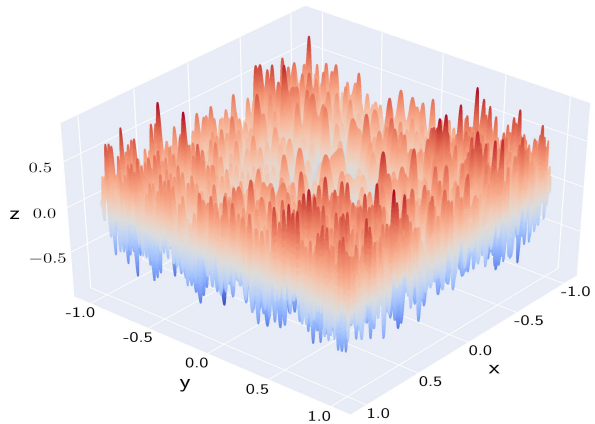
Experiment

$f(x_1, x_2) = \sum_{i=1}^{2} \sum_{j=1}^{2} a_{ij} \sin(sb_i x_i + sc_{i,j} x_i x_j) \cos(sb_j x_j + sd_{i,j} x_i^2), \quad s = 3.$
with 400×400 uniform data samples.
Network: (width, depth, rank)=(600, 15, 30).

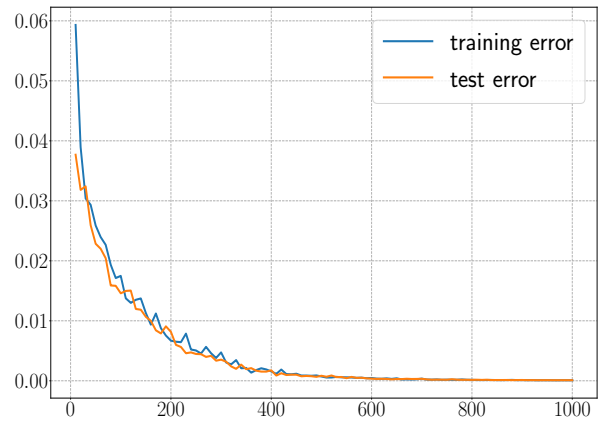
original function



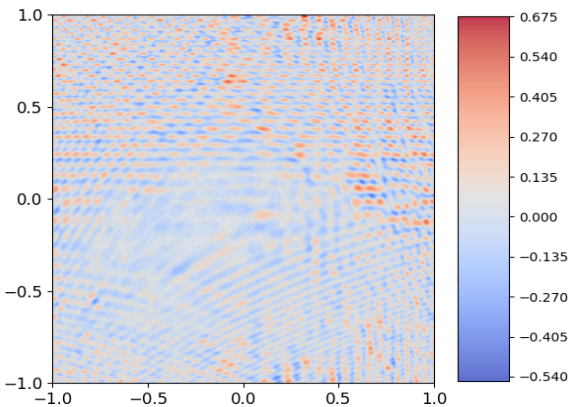
original function



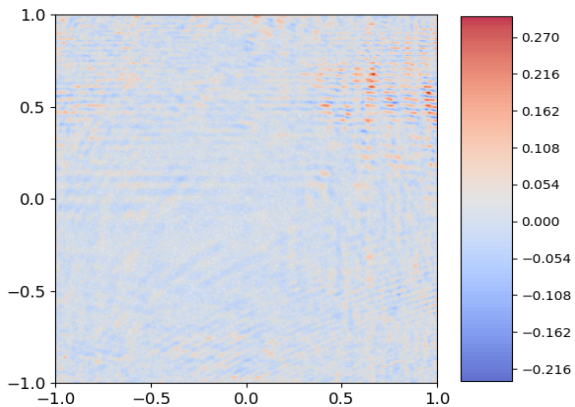
error in MSE



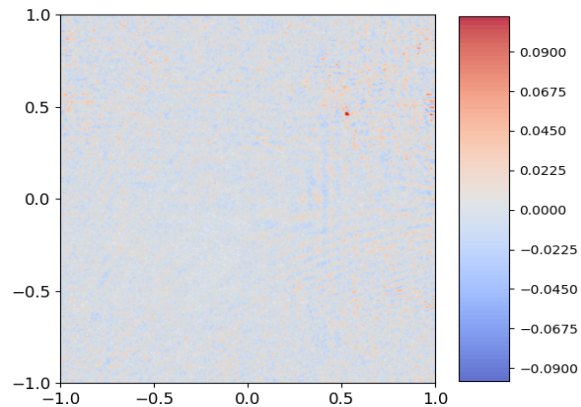
epoch 100



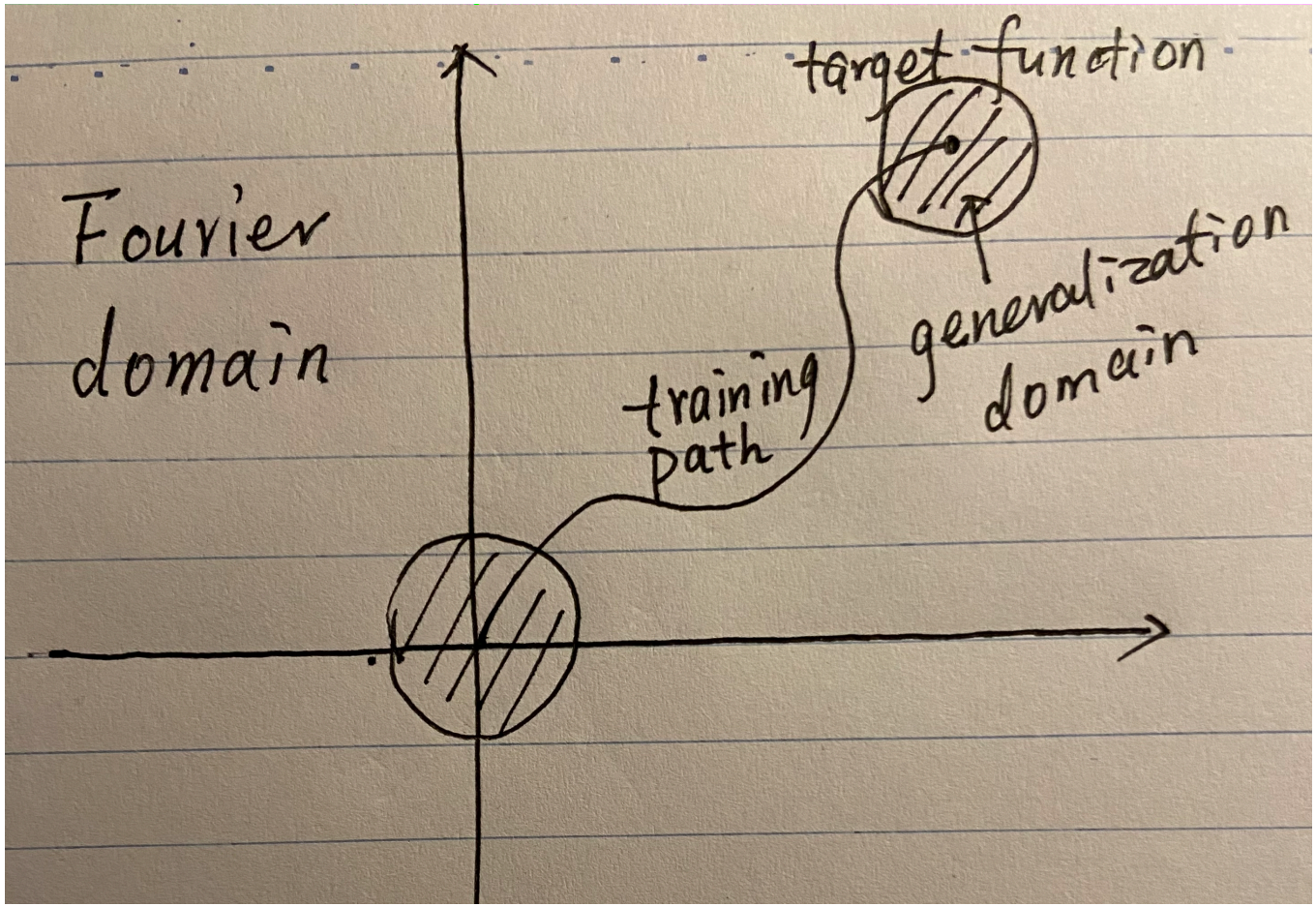
epoch 500



epoch 1000



Specifity vs. generality



More questions for multi-layer networks

- How to define the ”complexity” of a function for neural networks?
- How to characterize the approximation error and convergence?
- How to adjust network rank, width, and depth automatically?
- Understand the learning dynamics!