Lecture 10: Deep Network Approximation Preliminary and Barron Space

Haizhao Yang

Department of Mathematics
University of Maryland College Park

2022 Summer Mini Course
Tianyuan Mathematical Center in Central China

Supervised machine learning

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map f;
- Construct a finite family of maps $\{h(x;\theta)\}_\theta$;
- Create an empirical loss to quantify how good $h(x;\theta) \approx f(x)$ is:

$$R_S(\theta) := \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(h(x_i;\theta), y_i) \overset{\text{e.g.}}{=} \frac{1}{N}\sum_{i=1}^{N}(h(x_i;\theta) - y_i)^2;$$

- The best solution is $h(x;\theta_S)$ with

$$\theta_S = \operatorname{argmin} R_S(\theta);$$

- Use a numerical algorithm to solve the optimization problem and obtain a numerical solution $h(x;\theta_N)$.

Supervised machine learning

- Data $\{x_i\}_{i=1}^{n}$ are sampled randomly from an unknown distribution $U(x)$;
- Population loss as the ideal averaged prediction error quantification:
$$R_D(\theta) := \mathrm{E}_{x \sim U(\Omega)} \left[\mathcal{L}(h(x;\theta), f(x)) \right],$$
and the ideal prediction $h(x;\theta_D)$ with
$$\theta_D := \operatorname{argmin} R_D(\theta).$$
- In practice, $\theta_N \neq \theta_S \neq \theta_D$.
- How good does the actually learned function $h(x;\theta_N)$ predict $f(x)$ when x is unseen?
- $R_D(\theta_N)$ as the expected prediction error over all possible data samples.

Supervised machine learning

How large is the actual prediction error $R_D(\theta_N)$?

$$
\begin{aligned}
R_D(\theta_N) = & [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_N) - R_S(\theta_S)] + [R_S(\theta_S) - R_S(\theta_D)] \\
& + [R_S(\theta_D) - R_D(\theta_D)] + R_D(\theta_D) \\
\leq & \; {\color{blue}R_D(\theta_D)} + {\color{red}[R_S(\theta_N) - R_S(\theta_S)]} \\
& + [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_D) - R_D(\theta_D)],
\end{aligned}
$$

Supervised machine learning

How large is the actual prediction error $R_D(\theta_N)$?

$$\begin{aligned} R_D(\theta_N) &= [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_N) - R_S(\theta_S)] + [R_S(\theta_S) - R_S(\theta_D)] \\ &\quad + [R_S(\theta_D) - R_D(\theta_D)] + R_D(\theta_D) \\ &\le R_D(\theta_D) + [R_S(\theta_N) - R_S(\theta_S)] \\ &\quad + [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_D) - R_D(\theta_D)], \end{aligned}$$

- $R_D(\theta_D) = \int_\Omega (h(x;\theta_D) - f(x))^2 d\mu(x) \le= \int_\Omega (h(x;\tilde{\theta}) - f(x))^2 d\mu(x)$ can be bounded by a constructive approximation of $\tilde{\theta}$

Supervised machine learning

How large is the actual prediction error $R_D(\theta_N)$?

$$
\begin{aligned}
R_D(\theta_N) = {} & [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_N) - R_S(\theta_S)] + [R_S(\theta_S) - R_S(\theta_D)] \\
& + [R_S(\theta_D) - R_D(\theta_D)] + R_D(\theta_D) \\
\leq {} & R_D(\theta_D) + [R_S(\theta_N) - R_S(\theta_S)] \\
& + [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_D) - R_D(\theta_D)],
\end{aligned}
$$

- $R_D(\theta_D) = \int_\Omega (h(x;\theta_D) - f(x))^2 d\mu(x) \leq= \int_\Omega (h(x;\tilde{\theta}) - f(x))^2 d\mu(x)$ can be bounded by a constructive approximation of $\tilde{\theta}$
- $[R_S(\theta_N) - R_S(\theta_S)]$ is the optimization error

Supervised machine learning

How large is the actual prediction error $R_D(\theta_N)$?

$$\begin{aligned} R_D(\theta_N) &= [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_N) - R_S(\theta_S)] + [R_S(\theta_S) - R_S(\theta_D)] \\ &\quad + [R_S(\theta_D) - R_D(\theta_D)] + R_D(\theta_D) \\ &\le R_D(\theta_D) + [R_S(\theta_N) - R_S(\theta_S)] \\ &\quad + [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_D) - R_D(\theta_D)], \end{aligned}$$

- $R_D(\theta_D) = \int_\Omega (h(x;\theta_D) - f(x))^2 d\mu(x) \le= \int_\Omega (h(x;\tilde{\theta}) - f(x))^2 d\mu(x)$ can be bounded by a constructive approximation of $\tilde{\theta}$
- $[R_S(\theta_N) - R_S(\theta_S)]$ is the optimization error
- Other two terms are the generalization error

This lecture discusses the case when $h(x;\theta)$ is a deep neural network.

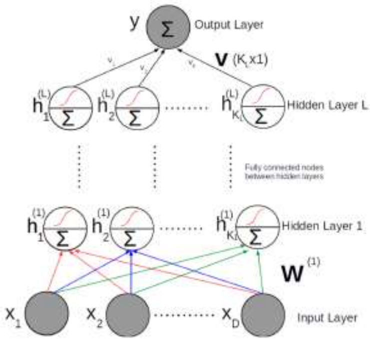
Deep learning

Function composition in the parametrization:

$$y = h(x;\theta) := T \circ \phi(x) := T \circ h^{(L)} \circ h^{(L-1)} \circ \cdots \circ h^{(1)}(x)$$

where

- $h^{(i)}(x) = \sigma({W^{(i)}}^T x + b^{(i)})$;
- $T(x) = V^T x$;
- $\theta = (W^{(1)}, \cdots, W^{(L)}, b^{(1)}, \cdots, b^{(L)}, V)$.



Foundation of deep learning

- Approximation theory: how good DNNs approximating functions?
- Optimization algorithms: how can we obtain (nearly) the best parameters?
- Generalization analysis: fixed noisy samples generalize?

This lecture focuses on the constructive approximation.

Analysis Goals and Applications

Goal

Given width N and depth L, what is the (optimal) approximation rate of DNNs for various function classes?

Why this goal?

In the optimization and generalization error analysis, the error is usually characterized in terms of width and depth

$$\begin{aligned} R_D(\theta_N) \leq R_D(\theta_D) &+ [R_S(\theta_N) - R_S(\theta_S)] \\ &+ [R_D(\theta_N) - R_S(\theta_N)] + [R_S(\theta_D) - R_D(\theta_D)], \end{aligned}$$

Schmidt-Hieber, 2017; Jacot et al., 2018; Mei et al., 2018; Cao and Gu, 2019; Chen et al., 2019b; Arora et al., 2019; Allen-Zhu et al., 2019; E et al., 2019; Ji and Telgarsky, 2020; etc.

Analysis Goals and Applications

Goal

Given width N and depth L, what is the (optimal) approximation rate of DNNs for various function classes?

Why this goal?

In scientific computing, a solver usually have two hyper-parameters N and L. For example, deep learning to solve PDEs (Han et al., Sirignano et al., Berg et al., Khoo et al., Maissi et al., etc.),

$$\begin{aligned} \mathcal{D}(u) &= f \quad \text{in } \Omega, \\ \mathcal{B}(u) &= g \quad \text{on } \partial\Omega. \end{aligned}$$

A DNN $\phi(\boldsymbol{x};\boldsymbol{\theta}^*)$ is constructed to approximate the solution $u(\boldsymbol{x})$ via

$$\begin{aligned} \boldsymbol{\theta}^* &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \\ &:= \operatorname*{argmin}_{\boldsymbol{\theta}} \|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\|_2^2 + \lambda\|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\|_2^2 \end{aligned}$$

Analysis Goals and Applications

Goals

- How good the approximation efficiency can be?
- The curse of dimensionality exist?

Why this goal?

- Computational efficiency
- For example, when we apply DNNs to solve high-dimensional PDEs, it is better to answer the above questions.

Literature Review

A long list with active research directions

- Cybenko, 1989; Hornik et al., 1989; Barron, 1993; Liang and Srikant, 2016; Yarotsky, 2017; Poggio et al., 2017; Schmidt-Hieber, 2017; E and Wang, 2018; Petersen and Voigtlaender, 2018; Chui et al., 2018; Yarotsky, 2018; Nakada and Imaizumi, 2019; Gribonval et al., 2019; Gühring et al., 2019; Chen et al., 2019; Li et al., 2019; Suzuki, 2019; Bao et al., 2019; E et al., 2019; Opschoor et al., 2019; Yarotsky and Zhevnerchuk, 2019; Bölcskei et al., 2019; Montanelli and Du, 2019; Chen and Wu, 2019; Zhou, 2020; Montanelli et al., 2020, etc.
- Function spaces: continuous functions, smooth functions, functions with integral representations;
- Tools: polynomial approximations, the law of large number, bit extraction technology (Bartlett et al., 1998; Harvey et al., 2017), Kolmogorov-Arnold representation theory, low-dimensional structures.

Our understanding on the literature

ReLU DNNs

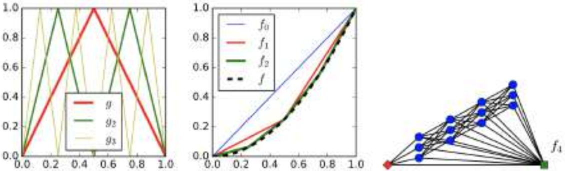
- Curse of dimensionality exisits for continuous and smooth functions.
- Exponential convergence is achievable for special function classes.

DNNs with advanced activation functions

- Curse of dimensionality does not exisit
- A small NN can be more powerful than you can expect

Topic 0: preliminary results

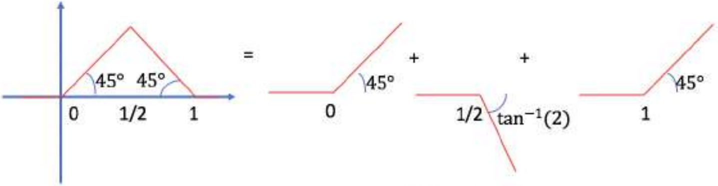
ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)



- Sawtooth function and compositions:

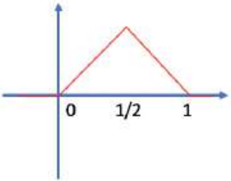
$$g(x) = \begin{cases} 2x, & x < \frac{1}{2} \\ 2(1-x), & x \geq \frac{1}{2} \end{cases} \quad \text{and} \quad g_s(x) = \underbrace{g \circ \cdots \circ g}_{s}(x).$$

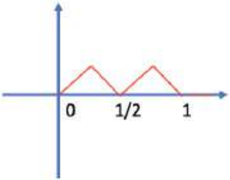
ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)



$$= \sigma(x) - \sigma\left(2\left(x - \tfrac{1}{2}\right)\right) + \sigma(x-1) = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} \sigma\left(\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix} \right) = W_2 \sigma(W_1 x + b) := g(x)$$

ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)





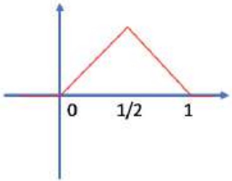
$g(x) = W_2\sigma(W_1 x + b)$ $\qquad$ $g_2(x) := g \circ g(x) = W_2\sigma(W_1 W_2 \sigma(W_1 x + b) + b)$

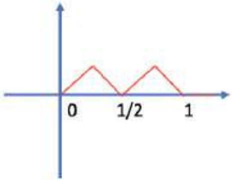
One-hidden-layer NN $\qquad$ Two-hidden-layer NN

Similarly, $g_s(x)$ is an s-hidden-layer NN with 2^s sawteeth

$$g_s(x) := g \circ \cdots \circ g(x) = W_2\sigma(W_1 W_2 \cdots \sigma(W_1 W_2 \sigma(W_1 W_2 \sigma(W_1 x + b) + b) + b) \cdots + b)$$

ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)





g(x)=$W_2\sigma(W_1x+b)$ $\qquad$ $g_2(x)$:= g∘g(x)=$W_2\sigma(W_1W_2\sigma(W_1x+b)+b)$

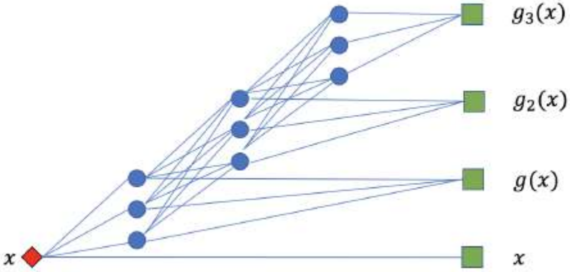
One-hidden-layer NN $\qquad$ Two-hidden-layer NN

Similarly, $g_s(x)$ is an s-hidden-layer NN with 2^s sawteeth

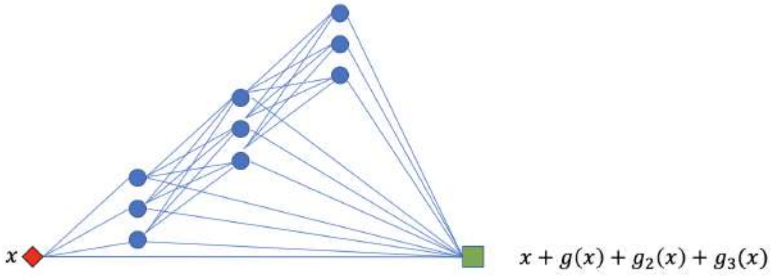
$g_s(x)$:= g∘ ⋯ ∘g(x)=$W_2\sigma(W_1W_2\cdots\sigma(W_1W_2\sigma(W_1W_2\sigma(W_1x+b)+b)+b)\cdots$+b)

$g_s(x)$ and $g_{s-1}(x)$ share the same first $s-2$ layers!

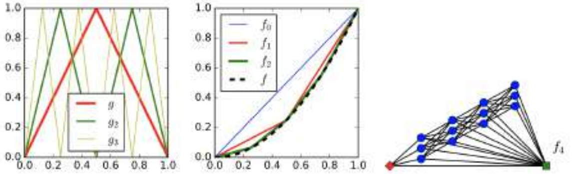
ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)



ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)



ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)

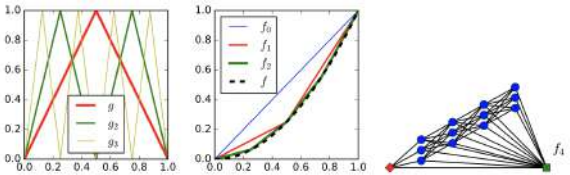


- Sawtooth function and compositions:

$$g(x) = \begin{cases} 2x, & x < \frac{1}{2} \\ 2(1-x), & x \geq \frac{1}{2} \end{cases} \quad \text{and} \quad g_s(x) = \underbrace{g \circ \cdots \circ g}_{s}(x).$$

- $f_L(x) = x - \sum_{s=1}^{L} \frac{g_s(x)}{2^{2s}} \approx x^2$ with an error $\epsilon = O(2^{-L})$.

ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)



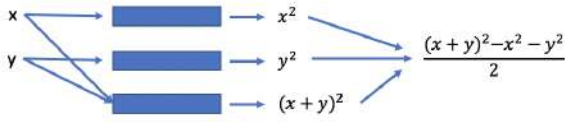
- Sawtooth function and compositions:

$$g(x) = \begin{cases} 2x, & x < \frac{1}{2} \\ 2(1-x), & x \geq \frac{1}{2} \end{cases} \quad \text{and} \quad g_s(x) = \underbrace{g \circ \cdots \circ g}_{s}(x).$$

- $f_L(x) = x - \sum_{s=1}^{L} \frac{g_s(x)}{2^{2s}} \approx x^2$ with an error $\epsilon = O(2^{-L})$.
- ReLU DNN $\approx x^2$, $L = W = O(\log(\frac{1}{\epsilon}))$.

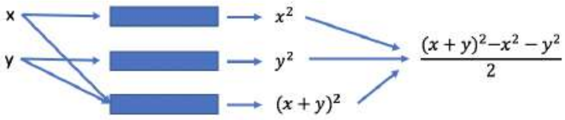
ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)

- ReLU DNN $\approx x^2$, $L = W = O(\log(\frac{1}{\epsilon}))$.
- $xy = \frac{(x+y)^2 - x^2 - y^2}{2}$, hence, ReLU DNN $\approx xy$, $L = W = O(\log(\frac{1}{\epsilon}))$.

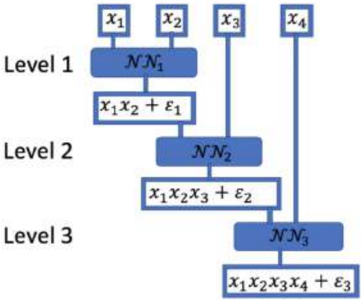


ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)

- ReLU DNN $\approx x^2$, $L = W = O(\log(\frac{1}{\epsilon}))$.
- $xy = \frac{(x+y)^2 - x^2 - y^2}{2}$, hence, ReLU DNN $\approx xy$, $L = W = O(\log(\frac{1}{\epsilon}))$.

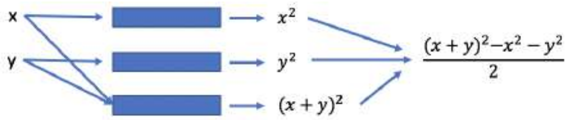


- ReLU DNN $\approx x_1 x_2 \ldots x_n$

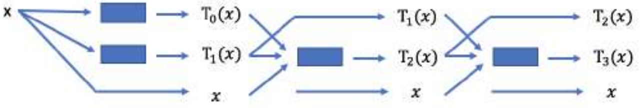


ReLU DNN $\approx x^2$ and polynomials (Yarosky, 2017)

- ReLU DNN $\approx x^2$, $L = W = O(\log(\frac{1}{\epsilon}))$.
- $xy = \frac{(x+y)^2 - x^2 - y^2}{2}$, hence, ReLU DNN $\approx xy$, $L = W = O(\log(\frac{1}{\epsilon}))$.



- ReLU DNN $\approx$ Chebyshev polynomial $T_n = 2xT_{n-1} - T_{n-2}$, $L = O(n \log \frac{n}{\epsilon} + n^2)$ and $W = O(n^2 \log \frac{n}{\epsilon} + n^2)$.



ReLU DNN$\approx x^2$ and polynomials (Lu, Shen, Y., Zhang, SIMA, 2021)

Lemma

For any $N, L \in \mathbb{N}^+$ and $a, b \in \mathbb{R}$ with $a < b$, there exists a ReLU FNN ϕ with width $9N + 1$ and depth L such that

$$|\phi(x, y) - xy| \le 6(b - a)^2 N^{-L}, \quad \text{for any } x, y \in [a, b].$$

Theorem

Assume $P(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$ for $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le k \in \mathbb{N}^+$. For any $N, L \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $9(N + 1) + k - 1$ and depth $7k^2 L$ such that

$$|\phi(\boldsymbol{x}) - P(\boldsymbol{x})| \le 9k(N + 1)^{-7kL}, \quad \text{for any } \boldsymbol{x} \in [0, 1]^d.$$

ReLU-ReLU^2 DNN reproduces x^2 and polynomials (Hon, Y., Neural Networks 2022)

Lemma

$f(x) = x^2$ can be realized exactly by a ReLU-ReLU^2 DNN with one hidden layer and two neurons.

Theorem

Assume $P(\boldsymbol{x}) = \sum_{j=1}^{J} c_j \boldsymbol{x}^{\boldsymbol{\alpha}_j}$ for $\boldsymbol{\alpha}_j \in \mathbb{N}^d$. For any $N, L, a, b \in \mathbb{N}^+$ such that $ab \geq J$ and $(L - 2b - b\log_2 N)N \geq b \max_j |\boldsymbol{\alpha}_j|$, there exists a ReLU-ReLU^2 DNN ϕ with width $4Na + 2d + 2$ and depth L such that

$$\phi(\boldsymbol{x}) = P(\boldsymbol{x}) \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d.$$

Topic 1: results by the law of large number theory

- Barron 1993
- Weinan E and students, 2019
- Chen, L. and Wu, C. 2019
- Montanelli, H., Yang, H., and Du, Q. 2020
- Siegel, J. and Xu, J., arXiv:2106.14997

Remark: It is argued that this class of functions is the natural class of functions of neural networks with stable numerical implementation and dimension-independent approximation rates.

Band-limited functions

Theorem (Montanelli, Y., Du, J. App. and Comp. Math. 2020)

Let $f : [0,1]^d \to \mathbb{R}$ *be a bandlimited function of the form*

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} F(\boldsymbol{w}) K(\boldsymbol{w} \cdot \boldsymbol{x}) d\boldsymbol{w}, \tag{1}$$

$$\operatorname{supp} F(\omega) \subset [-M, M]^d, \quad M \geq 1. \tag{2}$$

Suppose that K *is analytic and*

$$\int_{\mathbb{R}^d} |F(\boldsymbol{w})| d\boldsymbol{w} = \int_{[-M,M]^d} |F(\boldsymbol{w})| d\boldsymbol{w} = C_F < \infty. \tag{3}$$

Then there exists a deep ReLU network $\widetilde{f}(\boldsymbol{x})$ *of depth* $L = \mathcal{O}\left(\log_2^2 \frac{C_F}{\epsilon}\right)$ *and size* $W = \mathcal{O}\left(\frac{1}{\epsilon^2} \log_2^2 \frac{C_F}{\epsilon}\right)$ *such that*

$$\left\| \widetilde{f}(x) - f(x) \right\|_{L^2} \leq \epsilon. \tag{4}$$

Band-limited functions

Road map

- Monte Carlo:

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} F(\boldsymbol{w})K(\boldsymbol{w}\cdot\boldsymbol{x})d\boldsymbol{w} = \int_{\mathbb{R}^d} g(\boldsymbol{x},\boldsymbol{w})\frac{|F(\boldsymbol{w})|}{C_F}d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}}\left(g(\boldsymbol{x},\boldsymbol{w})\right).$$

$$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x},\boldsymbol{w}_j)$$

with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2.

- No curse of dimensionality by the law of large numbers.

Band-limited functions

Road map

- Monte Carlo:
$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} F(\boldsymbol{w})K(\boldsymbol{w}\cdot\boldsymbol{x})d\boldsymbol{w} = \int_{\mathbb{R}^d} g(\boldsymbol{x},\boldsymbol{w})\frac{|F(\boldsymbol{w})|}{C_F}d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}}\left(g(\boldsymbol{x},\boldsymbol{w})\right).$$
$$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x},\boldsymbol{w}_j)$$
with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2.
- No curse of dimensionality by the law of large numbers.
- ReLU DNN $\approx x^2$.
- ReLU DNN $\approx xy$.
- ReLU DNN $\approx$ (Chebyshev) polynomials.
- ReLU DNN $\approx$ analytic functions g.
- ReLU DNN $\approx f$, an automatic way to implement Monte Carlo.

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- Suppose supp $F(\omega) \subset [-M, M]^d$ and
$$\int_{\mathbb{R}^d} |F(\boldsymbol{w})| d\boldsymbol{w} = C_F < \infty.$$
- $$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} F(\boldsymbol{w}) K(\boldsymbol{w} \cdot \boldsymbol{x}) d\boldsymbol{w} = \int_{\mathbb{R}^d} g(\boldsymbol{x}, \boldsymbol{w}) \frac{|F(\boldsymbol{w})|}{C_F} d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}} \left(g(\boldsymbol{x}, \boldsymbol{w}) \right).$$

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- Suppose supp $F(\omega) \subset [-M, M]^d$ and
$$\int_{\mathbb{R}^d} |F(\boldsymbol{w})| d\boldsymbol{w} = C_F < \infty.$$
- $$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} F(\boldsymbol{w}) K(\boldsymbol{w} \cdot \boldsymbol{x}) d\boldsymbol{w} = \int_{\mathbb{R}^d} g(\boldsymbol{x}, \boldsymbol{w}) \frac{|F(\boldsymbol{w})|}{C_F} d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}} \left(g(\boldsymbol{x}, \boldsymbol{w}) \right).$$
- f is a convex combination of functions in a $L^2([0,1]^d)$-bounded set
$$G = \{\gamma \left[\cos(\beta) K_r(\boldsymbol{w} \cdot \boldsymbol{x}) - \sin(\beta) K_i(\boldsymbol{w} \cdot \boldsymbol{x}) \right], \beta \in \mathbb{R}, |\gamma| \leq C_F \}.$$

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- Suppose supp $F(\omega) \subset [-M, M]^d$ and
$$\int_{\mathbb{R}^d} |F(\boldsymbol{w})| d\boldsymbol{w} = C_F < \infty.$$
- $$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} F(\boldsymbol{w}) K(\boldsymbol{w} \cdot \boldsymbol{x}) d\boldsymbol{w} = \int_{\mathbb{R}^d} g(\boldsymbol{x}, \boldsymbol{w}) \frac{|F(\boldsymbol{w})|}{C_F} d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}} \left(g(\boldsymbol{x}, \boldsymbol{w}) \right).$$
- f is a convex combination of functions in a $L^2([0,1]^d)$-bounded set
$$G = \{\gamma \left[\cos(\beta) K_r(\boldsymbol{w} \cdot \boldsymbol{x}) - \sin(\beta) K_i(\boldsymbol{w} \cdot \boldsymbol{x}) \right], \ \beta \in \mathbb{R}, \ |\gamma| \leq C_F\}.$$
- Monte Carlo:
$$f(x) \approx f_n(\boldsymbol{x}) = \sum_{j=1}^{n} a_j K_r(\boldsymbol{w}_j \cdot \boldsymbol{x}) + b_j K_i(\boldsymbol{w}_j \cdot \boldsymbol{x})$$
$$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n} \sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j)$$
with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2.

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j K(\boldsymbol{w}_j \cdot \boldsymbol{x})$$

  with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2 and no curse of dimensionality, i.e. $n = (\frac{1}{\epsilon})^2 \ll (\frac{1}{\epsilon})^d$.

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j K(\boldsymbol{w}_j \cdot \boldsymbol{x})$$
  with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2 and no curse of dimensionality, i.e. $n = (\frac{1}{\epsilon})^2 \ll (\frac{1}{\epsilon})^d$.
- $K(x) \approx \sum_{k=0}^{m} c_k T_k(x)$ with error $\epsilon = O(e^{-cm})$ i.e. $m = O(\log(\frac{1}{\epsilon}))$.

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j K(\boldsymbol{w}_j \cdot \boldsymbol{x})$$

  with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2 and no curse of dimensionality, i.e. $n = (\frac{1}{\epsilon})^2 \ll (\frac{1}{\epsilon})^d$.
- $K(x) \approx \sum_{k=0}^{m} c_k T_k(x)$ with error $\epsilon = O(e^{-cm})$ i.e. $m = O(\log(\frac{1}{\epsilon}))$.
- $O(mn)$ Chebyshev polynomials with degree bounded by m needed to approximate $f(x)$.

$$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j \sum_{k=0}^{m} c_k T_k(\boldsymbol{w}_j \cdot \boldsymbol{x})$$

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j K(\boldsymbol{w}_j \cdot \boldsymbol{x})$$
  with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2 and no curse of dimensionality, i.e. $n = (\frac{1}{\epsilon})^2 \ll (\frac{1}{\epsilon})^d$.
- $K(x) \approx \sum_{k=0}^{m} c_k T_k(x)$ with error $\epsilon = O(e^{-cm})$ i.e. $m = O(\log(\frac{1}{\epsilon}))$.
- $O(mn)$ Chebyshev polynomials with degree bounded by m needed to approximate $f(x)$.
  $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j \sum_{k=0}^{m} c_k T_k(\boldsymbol{w}_j \cdot \boldsymbol{x})$$
- ReLU DNN $\approx$ Chebyshev polynomial T_k, $L = O(k \log \frac{k}{\epsilon} + k^2)$ and $W = O(k^2 \log \frac{k}{\epsilon} + k^2)$.

ReLU DNN $\approx$ band-limited functions (Montanelli, Y., Du, JCM, 2020)

- $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j K(\boldsymbol{w}_j \cdot \boldsymbol{x})$$
  with an error $\epsilon = O(\frac{1}{\sqrt{n}})$ in L^2 and no curse of dimensionality, i.e. $n = (\frac{1}{\epsilon})^2 \ll (\frac{1}{\epsilon})^d$.
- $K(x) \approx \sum_{k=0}^{m} c_k T_k(x)$ with error $\epsilon = O(e^{-cm})$ i.e. $m = O(\log(\frac{1}{\epsilon}))$.
- $O(mn)$ Chebyshev polynomials with degree bounded by m needed to approximate $f(x)$.
  $$f(x) \approx f_n(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} g(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{j=1}^{n} a_j \sum_{k=0}^{m} c_k T_k(\boldsymbol{w}_j \cdot \boldsymbol{x})$$
- ReLU DNN $\approx$ Chebyshev polynomial T_k, $L = O(k \log \frac{k}{\epsilon} + k^2)$ and $W = O(k^2 \log \frac{k}{\epsilon} + k^2)$.
- Total complexity DNN is $O(mn(m^2 \log \frac{m}{\epsilon} + m^2))$, a polynomial of $O(\frac{1}{\epsilon})$ and no curse of dimensionality.