Numerical Neural Network

Jinchao Xu

Penn State University

xu@math.psu.edu   http://www.math.psu.edu/xu/

NEMESIS, June 14, 2021

(NEw generation MEthods for numerical SImulationS)

NSF: DMS-1819157

Finite element: Piecewise linear functions

Finite element: Piecewise linear functions

- Uniform grid $\mathcal{T}_h$

$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \; (j = 0 : N+1).$$

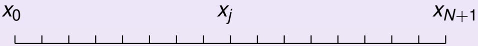


Figure: 1D uniform grid

Finite element: Piecewise linear functions

- Uniform grid $\mathcal{T}_h$

$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \ (j = 0 : N+1).$$

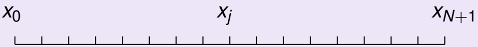
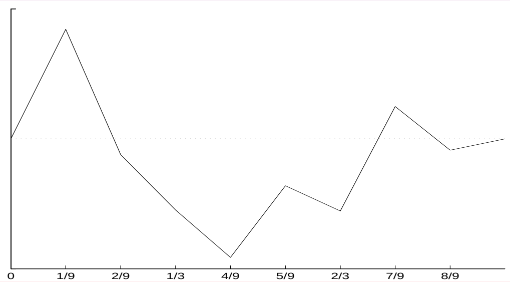


Figure: 1D uniform grid

- Linear finite element space

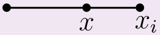
$$V_h = \{v : \ v \text{ is continuous and piecewise linear w.r.t. } \mathcal{T}_h\}.$$

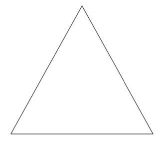


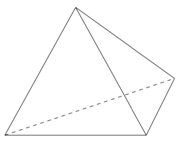
Finite element in multi-dimensions

$(k = 1)$

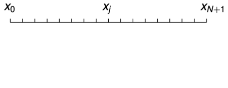
$w_1 x + b$ $\qquad$ $w_1 x_1 + w_2 x_2 + b$ $\qquad$ $w_1 x_1 + w_2 x_2 + w_3 x_3 + b$ $\quad \cdots$

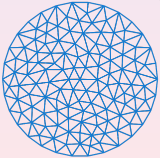


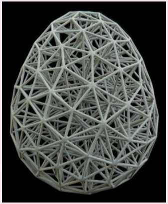




$\cdots$





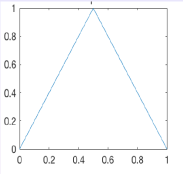


$\cdots$

FEM basis function in 1D

- Denote the basis function in $\mathcal{T}_1$

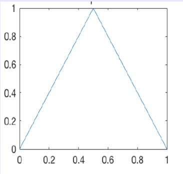
$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}. \tag{1}$$

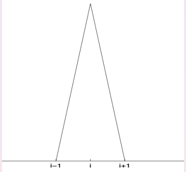


FEM basis function in 1D

- Denote the basis function in $\mathcal{T}_1$

$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}. \tag{1}$$





- All basis functions φ_i can be written as

$$\varphi_i = \varphi(\frac{x - x_{i-1}}{2h}) = \varphi(w_h x + b_i). \tag{2}$$

with $w_h = \frac{1}{2h}$, $\quad b_i = \frac{-(i-1)}{2}$.

FEM basis function in 1D

- Denote the basis function in $\mathcal{T}_1$

$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}. \tag{1}$$

- All basis functions φ_i can be written as

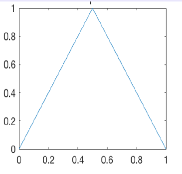
$$\varphi_i = \varphi(\frac{x - x_{i-1}}{2h}) = \varphi(w_h x + b_i). \tag{2}$$

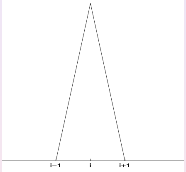
with $w_h = \frac{1}{2h}$, $\quad b_i = \frac{-(i-1)}{2}$.

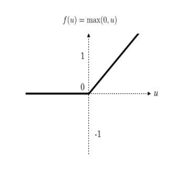
- Let $x_+ = \max(0, x) = \mathrm{ReLU}(x)$,

$$\varphi(x) = 2x_+ - 4(x - 1/2)_+ + 2(x-1)_+.$$

- $\varphi_i \in \mathrm{span}\left\{(wx+b)_+, w, b \in \mathbb{R}^1\right\}$







FEM $\Longrightarrow \Sigma_n^1$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

FEM $\Longrightarrow \Sigma_n^1$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

$$\text{FEM} \subset \operatorname{span}\left\{(wx+b)_+, w, b \in \mathbb{R}^1\right\} = \Sigma_n^1.$$

FEM $\Longrightarrow \Sigma_n^1$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

$$\text{FEM} \subset \operatorname{span}\left\{(wx+b)_+, w, b \in \mathbb{R}^1\right\} = \Sigma_n^1.$$

- Example:

$$f \in \operatorname{span}\left\{(wx+b)_+,\ w, b \in \mathbb{R}^1\right\} \longleftrightarrow f = \sum_{j=1}^{n} a_j (w_j x + b_j)_+. \tag{3}$$

FEM $\Longrightarrow \Sigma_n^1$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

$$\text{FEM} \subset \operatorname{span}\left\{(wx+b)_+, w, b \in \mathbb{R}^1\right\} = \Sigma_n^1.$$

- Example:

$$f \in \operatorname{span}\left\{(wx+b)_+,\ w, b \in \mathbb{R}^1\right\} \longleftrightarrow f = \sum_{j=1}^{n} a_j (w_j x + b_j)_+. \tag{3}$$

f is one hidden layer "deep" neural network with activation function ReLU, n neurons.

Generalization to multi-dimension:

Higher dimension $d \geq 1$

$$\Sigma_n^1 = \left\{ \sum_{i=1}^{n} a_i(\omega_i \cdot x + b_i)_+ : \omega_i \in \mathbb{R}^d, b_i \in \mathbb{R} \right\} \tag{4}$$

Generalization to multi-dimension:

Higher dimension $d \geq 1$

$$\Sigma_n^1 = \left\{ \sum_{i=1}^{n} a_i(\omega_i \cdot x + b_i)_+ : \omega_i \in \mathbb{R}^d, b_i \in \mathbb{R} \right\} \tag{4}$$

Shallow Neural Network: General activation function: $\sigma : \mathbb{R}^1 \mapsto \mathbb{R}^1$, namely

$$\Sigma_n^\sigma = \left\{ \sum_{i=1}^{n} a_i\sigma(\omega_i \cdot x + b_i) : \omega_i \in \mathbb{R}^d, b_i \in \mathbb{R} \right\} \tag{5}$$

Generalization to multi-dimension:

Higher dimension $d \geq 1$

$$\Sigma_n^1 = \left\{ \sum_{i=1}^{n} a_i(\omega_i \cdot x + b_i)_+ : \omega_i \in \mathbb{R}^d, b_i \in \mathbb{R} \right\} \tag{4}$$

Shallow Neural Network: General activation function: $\sigma : \mathbb{R}^1 \mapsto \mathbb{R}^1$, namely

$$\Sigma_n^\sigma = \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i) : \omega_i \in \mathbb{R}^d, b_i \in \mathbb{R} \right\} \tag{5}$$

Common activation functions:

- Heaviside $\sigma = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$
- Sigmoidal $\sigma = (1 + e^{-x})^{-1}$
- Rectified Linear with $\sigma = \max(0, x)$
- Power of a ReLU $\sigma = [\max(0, x)]^k$
- Cosine $\sigma = \cos(x)$

σ-DNN: Linears, activation and composition

σ-DNN: Linears, activation and composition

1. Start from a linear function

$$W^0 x + b^0$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0 x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0 x + b^0)$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0 x + b^0$$

2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0 x + b^0)$$

3. Compose with another linear function:
$$W^1 x^{(1)} + b^1$$

4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1 x^{(1)} + b^1)$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x;\Theta) = W^2x^{(2)} + b^2$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x;\Theta) = W^2x^{(2)} + b^2$$
6. ...

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x;\Theta) = W^2x^{(2)} + b^2$$
6. ...

Deep neural network functions with ℓ-hidden layers

$$\Sigma^{\sigma}_{n_{1:\ell}} = \{W^{\ell}x^{(\ell)} + b^{\ell},\ W^i \in \mathbb{R}^{n_i},\ b_i \in \mathbb{R}\}$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x;\Theta) = W^2x^{(2)} + b^2$$
6. ...

Deep neural network functions with ℓ-hidden layers

$$\Sigma^{\sigma}_{n_{1:\ell}} = \{W^{\ell}x^{(\ell)} + b^{\ell},\ W^i \in \mathbb{R}^{n_i},\ b_i \in \mathbb{R}\}$$

$$\Sigma^{k}_{n_{1:\ell}} = \Sigma^{ReLU^k}_{n_{1:\ell}}.$$

What does a function in ReLU-DNN look like?

Obviously:

$$\Sigma^1_{n_{1:\ell}} = \text{a space of continuous piecewise linear functions!}$$

What does a function in ReLU-DNN look like?

Obviously:

$$\Sigma^1_{n_{1:\ell}} = \text{a space of continuous piecewise linear functions!}$$

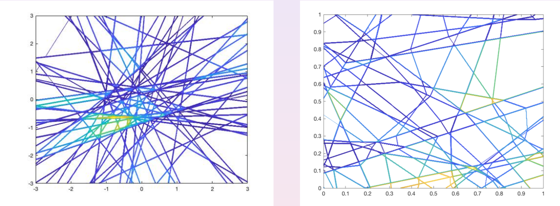


Figure: $\ell = 1$ and $\ell = 2$

How is $\mathcal{N}_\ell^1$ compared with (adaptive) linear FEM?

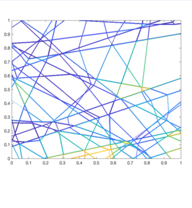


Figure: $(40, 40)$

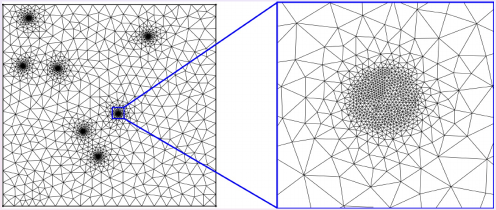


Figure: Adaptive Grid

Connection of ReLU DNN and Linear FEM

1. $d = 1$,

$$\mathrm{FE} \subset \Sigma_n^1.$$

Connection of ReLU DNN and Linear FEM

1. $d = 1$,
$$\mathrm{FE} \subset \Sigma_n^1.$$
2. $d \geq 2$,
$$\mathrm{FE} \not\subseteq \Sigma_n^1.$$

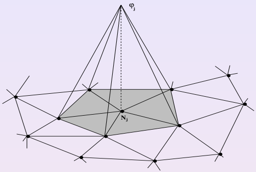
Connection of ReLU DNN and Linear FEM

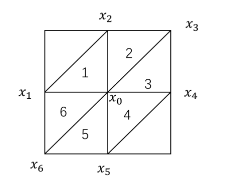
1. $d = 1$,
$$\mathrm{FE} \subset \Sigma_n^1.$$
2. $d \geq 2$,
$$\mathrm{FE} \not\subseteq \Sigma_n^1.$$
3. $d \geq 2$,
$$\mathrm{FE} \subset \Sigma_{n_{1:\ell}}^1 \quad \text{for some } \ell > 1.$$

Refs: R. Arora, A. Basu, P. Mianjy & A. Mukherjee, 2016, J. He, L. Li, J. Xu & C. Zheng, 2018

A 2D example: FE basis function

Consider a 2D FE basis function, $\phi(x)$:

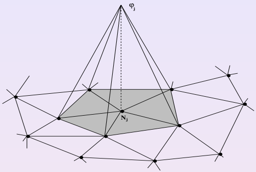


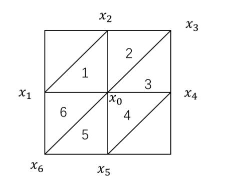


[1] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527, Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

A 2D example: FE basis function

Consider a 2D FE basis function, $\phi(x)$:





Here g_i is linear in Domain i, and $x_7 = x_1$, satisfying

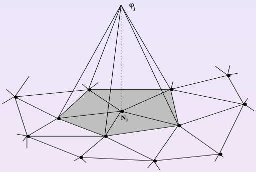
$$g_i(x_0) = 1 \qquad g_i(x_i) = 0 \qquad g_i(x_{i+1}) = 0$$

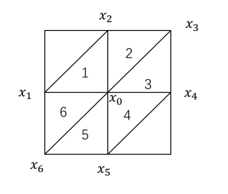
$$\phi(x) = \begin{cases} g_i(x), & x \in \text{Domain } i \\ 0. & x \in \mathbb{R}^2 - \overline{x_1 x_2 x_3 x_4 x_5 x_6} \end{cases} \tag{6}$$

[1] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527, Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

A 2D example: FE basis function

Consider a 2D FE basis function, $\phi(x)$:





Here g_i is linear in Domain i, and $x_7 = x_1$, satisfying

$$g_i(x_0) = 1 \qquad g_i(x_i) = 0 \qquad g_i(x_{i+1}) = 0$$

$$\phi(x) = \begin{cases} g_i(x), & x \in \text{Domain } i \\ 0. & x \in \mathbb{R}^2 - \overline{x_1 x_2 x_3 x_4 x_5 x_6} \end{cases} \tag{6}$$

It is non-obvious, but in fact we have[1]

$$\phi(x) \in \mathrm{DNN}_2(\mathrm{ReLU}) \tag{7}$$

[1] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527, Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

ReLU-DNN and Linear FEM for H^1

$$\mathit{ReLU\text{-}DNN} = \Sigma^1_{n_{1:\ell}}$$

ReLU-DNN and Linear FEM for H^1

$$\textit{ReLU-DNN} = \Sigma^1_{n_{1:\ell}} = \textit{Linear FEM} \subset H^1(\Omega)$$

Basic Approximation Properties

Can shallow networks approximate arbitrary functions?

Basic Approximation Properties

Can shallow networks approximate arbitrary functions?
Let

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

Basic Approximation Properties

Can shallow networks approximate arbitrary functions?
Let

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Is

$$\bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{8}$$

  dense in $L^2(\Omega)$ or C^k?

Basic Approximation Properties

Can shallow networks approximate arbitrary functions?
Let

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^n a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Is

$$\bigcup_{n=1}^\infty \Sigma_n^\sigma \tag{8}$$

  dense in $L^2(\Omega)$ or C^k?
- Yes, if and only if σ is NOT a polynomial!

Basic Approximation Properties

Can shallow networks approximate arbitrary functions?
Let

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Is

$$\bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{8}$$

  dense in $L^2(\Omega)$ or C^k?
- Yes, if and only if σ is NOT a polynomial!
- Our interest: When can this approximation be done in a stable manner?

Stable Neural Network Approximation

- Consider approximation from the class

$$\Sigma_{n,M}^{\sigma} := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \sum_{i=1}^{n} |a_i| \leq M \right\} \tag{9}$$

of neural networks with ℓ^1-bounded outer coefficients.

Stable Neural Network Approximation

- Consider approximation from the class

$$\Sigma^{\sigma}_{n,M} := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \sum_{i=1}^{n} |a_i| \le M \right\} \tag{9}$$

  of neural networks with ℓ^1-bounded outer coefficients.

- More generally for a dictionary $\mathbb{D} \subset H = L^2(\Omega)$, consider

$$\Sigma_{n,M}(\mathbb{D}) = \left\{ \sum_{i=1}^{n} a_i h_i,\ h_i \in \mathbb{D},\ \sum_{i=1}^{n} |a_i| \le M \right\} \tag{10}$$

Stable Neural Network Approximation

- Consider approximation from the class

$$\Sigma^{\sigma}_{n,M} := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \sum_{i=1}^{n} |a_i| \leq M \right\} \tag{9}$$

  of neural networks with ℓ^1-bounded outer coefficients.
- More generally for a dictionary $\mathbb{D} \subset H = L^2(\Omega)$, consider

$$\Sigma_{n,M}(\mathbb{D}) = \left\{ \sum_{i=1}^{n} a_i h_i,\ h_i \in \mathbb{D},\ \sum_{i=1}^{n} |a_i| \leq M \right\} \tag{10}$$

- Let $M < \infty$ be fixed and consider approximation as $n \to \infty$.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[2]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\left\{ \sum_{j=1}^{n} a_j h_j : \; n \in \mathbb{N}, \; h_j \in \mathbb{D}, \; \sum_{i=1}^{n} |a_i| \leq 1 \right\}}, \tag{11}$$

[2] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[2]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\left\{ \sum_{j=1}^{n} a_j h_j : \ n \in \mathbb{N}, \ h_j \in \mathbb{D}, \ \sum_{i=1}^{n} |a_i| \leq 1 \right\}}, \tag{11}$$

- Define a norm

$$\|f\|_{\mathcal{K}_1(\mathbb{D})} = \inf\{r > 0 : f \in rB_1(\mathbb{D})\}, \tag{12}$$

as the guage of the set $B_1(\mathbb{D})$.

[2] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[2]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\left\{ \sum_{j=1}^{n} a_j h_j : \ n \in \mathbb{N}, \ h_j \in \mathbb{D}, \ \sum_{i=1}^{n} |a_i| \leq 1 \right\}}, \tag{11}$$

- Define a norm

$$\|f\|_{\mathcal{K}_1(\mathbb{D})} = \inf\{r > 0 : f \in rB_1(\mathbb{D})\}, \tag{12}$$

as the guage of the set $B_1(\mathbb{D})$.

- The unit ball is

$$\{f \in H : \|f\|_{\mathcal{K}_1(\mathbb{D})} \leq 1\} = B_1(\mathbb{D}). \tag{13}$$

[2] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[2]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\left\{\sum_{j=1}^{n} a_j h_j :\ n \in \mathbb{N},\ h_j \in \mathbb{D},\ \sum_{i=1}^{n} |a_i| \le 1\right\}}, \tag{11}$$

- Define a norm

$$\|f\|_{\mathcal{K}_1(\mathbb{D})} = \inf\{r > 0 : f \in rB_1(\mathbb{D})\}, \tag{12}$$

as the guage of the set $B_1(\mathbb{D})$.

- The unit ball is

$$\{f \in H : \|f\|_{\mathcal{K}_1(\mathbb{D})} \le 1\} = B_1(\mathbb{D}). \tag{13}$$

- We have

$$\{f \in H : \|f\|_{\mathcal{K}_1(\mathbb{D})} < \infty\} \tag{14}$$

is a Banach space.

[2] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Example: $H = \ell^2$

- Let $H = \ell^2$, $\mathbb{D} = \{e_1, e_2, ...\}$.
- What is $B_1(\mathbb{D})$?

Example: $H = \ell^2$

- Let $H = \ell^2$, $\mathbb{D} = \{e_1, e_2, ...\}$.
- What is $B_1(\mathbb{D})$?
- The convex hull of $\pm\mathbb{D}$ is

$$B_1(\mathbb{D}) = \{(a_1, a_2, ...) \in \ell^2 : \sum_{i=1}^{\infty} |a_i| \leq 1\} \tag{15}$$

Example: $H = \ell^2$

- Let $H = \ell^2$, $\mathbb{D} = \{e_1, e_2, ...\}$.
- What is $B_1(\mathbb{D})$?
- The convex hull of $\pm\mathbb{D}$ is

$$B_1(\mathbb{D}) = \{(a_1, a_2, ...) \in \ell^2 : \sum_{i=1}^{\infty} |a_i| \leq 1\} \tag{15}$$

- Thus the norm is given by

$$\mathcal{K}_1(\mathbb{D}) = \ell^1 \subset \ell^2. \tag{16}$$

Stable Dictionary Approximation Space

Theorem (Siegel & Xu 2021)

A function $f \in H = L^2(\Omega)$ can be approximated at all, i.e.

$$\lim_{n\to\infty} \inf_{f_n \in \Sigma_{n,M}(\mathbb{D})} \|f - f_n\|_H = 0, \tag{17}$$

for a fixed $M < \infty$ if and only if

$$f \in MB_1(\mathbb{D}) \subset \mathcal{K}_1(\mathbb{D}).$$

Stable Dictionary Approximation Space

Theorem (Siegel & Xu 2021)

A function $f \in H = L^2(\Omega)$ can be approximated at all, i.e.

$$\lim_{n\to\infty} \inf_{f_n \in \Sigma_{n,M}(\mathbb{D})} \|f - f_n\|_H = 0, \tag{17}$$

for a fixed $M < \infty$ if and only if

$$f \in MB_1(\mathbb{D}) \subset \mathcal{K}_1(\mathbb{D}).$$

Furthermore, if

$$\|\mathbb{D}\| \equiv \sup_{h\in\mathbb{D}} \|h\|_H < \infty$$

we have

$$\inf_{f_n \in \Sigma_{n,M}(\mathbb{D})} \|f - f_n\|_H \le n^{-\frac{1}{2}} \|\mathbb{D}\| \|f\|_{\mathcal{K}_1(\mathbb{D})}. \tag{18}$$

The Spectral Barron Space

- Let $f \in B_1(\mathbb{D})$, $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{F}_s^d := \{(1 + |\omega|)^{-s} e^{2\pi i \omega \cdot x} : \omega \in \mathbb{R}^d\} \tag{19}$$

[3] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

The Spectral Barron Space

- Let $f \in B_1(\mathbb{D})$, $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{F}_s^d := \{(1 + |\omega|)^{-s} e^{2\pi i \omega \cdot x} \; : \; \omega \in \mathbb{R}^d\} \tag{19}$$

- In this case the norm is characterized by[3]

$$\|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} = \inf_{f_e|_{B_1^d} = f} \int_{\mathbb{R}^d} (1 + |\xi|)^s |\hat{f}_e(\xi)| d\xi, \tag{20}$$

where the infimum is taken over all extensions $f_e \in L^1(\mathbb{R}^d)$.

[3] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

The Spectral Barron Space

- Let $f \in B_1(\mathbb{D})$, $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \le 1\}$, and

$$\mathbb{D} = \mathbb{F}_s^d := \{(1+|\omega|)^{-s} e^{2\pi i \omega \cdot x} \,:\, \omega \in \mathbb{R}^d\} \tag{19}$$

- In this case the norm is characterized by[3]

$$\|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} = \inf_{f_e|_{B_1^d} = f} \int_{\mathbb{R}^d} (1+|\xi|)^s |\hat{f}_e(\xi)| d\xi, \tag{20}$$

  where the infimum is taken over all extensions $f_e \in L^1(\mathbb{R}^d)$.
- Property:

$$H^{s+\frac{d}{2}+\varepsilon}(\Omega) \hookrightarrow B^s(\Omega) \hookrightarrow W^{s,\infty}(\Omega). \tag{21}$$

[3] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

The Barron Space

The results are proved in Siegel and Xu 2021[4]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{22}$$

where $\sigma_k = [\max(0, x)]^k$.

[4] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[5] W. E, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[6] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

The Barron Space

The results are proved in Siegel and Xu 2021[4]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{22}$$

  where $\sigma_k = [\max(0, x)]^k$.
- When $k = 1$, $\mathcal{K}_1(\mathbb{P}_k^d)$ is equivalentto the Barron space (introduced in[5]).

---

[4] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[5] W. E, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[6] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

The Barron Space

The results are proved in Siegel and Xu 2021[4]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{22}$$

  where $\sigma_k = [\max(0, x)]^k$.
- When $k = 1$, $\mathcal{K}_1(\mathbb{P}_k^d)$ is equivalentto the Barron space (introduced in[5]).
- When $k = 0$, $d = 1$, $\mathcal{K}_1(\mathbb{P}_k^d) = BV([-1, 1])$.

[4] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[5] W. E, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[6] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

The Barron Space

The results are proved in Siegel and Xu 2021[4]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \le 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{22}$$

  where $\sigma_k = [\max(0, x)]^k$.
- When $k = 1$, $\mathcal{K}_1(\mathbb{P}_k^d)$ is equivalentto the Barron space (introduced in[5]).
- When $k = 0$, $d = 1$, $\mathcal{K}_1(\mathbb{P}_k^d) = BV([-1, 1])$.
- We have$\mathcal{K}_1(\mathbb{P}_k^d) \supset \mathcal{K}_1(\mathbb{F}_{k+1}^d)$ (for $k = 0$, Barron 1993[6])

[4] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[5] W. E, Chao Ma, and Lei Wu. “Barron spaces and the compositional function spaces for neural network models”. In: *arXiv preprint arXiv:1906.08039* (2019).

[6] Andrew R Barron. “Universal approximation bounds for superpositions of a sigmoidal function”. In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

Previous State-of-the-art Results

For some dictionaries $\mathbb{D}$, the $n^{-\frac{1}{2}}$ approximation rate can be improved!

- For $\mathbb{D} = \mathbb{P}_0^d$, we have[7]

$$\sup_{f\in B_1(\mathbb{D})} \inf_{f_n\in\Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2}-\frac{1}{2d}}. \tag{23}$$

- For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have[8],[9], if f is in some spectral Barron space:

$$\inf_{f_n\in\Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2}-\frac{1}{d}}. \tag{24}$$

[7] Yuly Makovoz. “Random approximants and neural networks”. In: *Journal of Approximation Theory* 85.1 (1996), pp. 98–109.

[8] Jason M Klusowski and Andrew R Barron. “Approximation by Combinations of ReLU and Squared ReLU Ridge Functions With ℓ^1 and ℓ^0 Controls”. In: *IEEE Transactions on Information Theory* 64.12 (2018), pp. 7649–7656.

[9] Jinchao Xu. “Finite Neuron Method and Convergence Analysis”. In: *Communications in Computational Physics* 28.5 (2020), pp. 1707–1745.

Previous State-of-the-art Results

For some dictionaries $\mathbb{D}$, the $n^{-\frac{1}{2}}$ approximation rate can be improved!

- For $\mathbb{D} = \mathbb{P}_0^d$, we have[7]

$$\sup_{f \in B_1(\mathbb{D})} \inf_{f_n \in \Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2} - \frac{1}{2d}}. \tag{23}$$

- For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have[8],[9], if f is in some spectral Barron space:

$$\inf_{f_n \in \Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2} - \frac{1}{d}}. \tag{24}$$

- What are the optimal approximation rates?

---

[7] Yuly Makovoz. "Random approximants and neural networks". In: *Journal of Approximation Theory* 85.1 (1996), pp. 98–109.

[8] Jason M Klusowski and Andrew R Barron. "Approximation by Combinations of ReLU and Squared ReLU Ridge Functions With ℓ^1 and ℓ^0 Controls". In: *IEEE Transactions on Information Theory* 64.12 (2018), pp. 7649–7656.

[9] Jinchao Xu. "Finite Neuron Method and Convergence Analysis". In: *Communications in Computational Physics* 28.5 (2020), pp. 1707–1745.

New Optimal Bounds[10]

Theorem

For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have

$$n^{-\frac{1}{2}-\frac{2k+1}{2d}} \lesssim \sup_{f\in B_1(\mathbb{D})} \inf_{f_n\in\Sigma_{n,M}} \|f - f_n\|_{L^2(\Omega)} \lesssim n^{-\frac{1}{2}-\frac{2k+1}{2d}} \tag{25}$$

[10] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

New Optimal Bounds[10]

Theorem

For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have

$$n^{-\frac{1}{2}-\frac{2k+1}{2d}} \lesssim \sup_{f\in B_1(\mathbb{D})} \inf_{f_n\in\Sigma_{n,M}} \|f - f_n\|_{L^2(\Omega)} \lesssim n^{-\frac{1}{2}-\frac{2k+1}{2d}} \tag{25}$$

In comparison: optimal bound for finite elements

Theorem

Assume that V_h^k is a finite element of degree k on quasi-uniform mesh $\{\mathcal{T}_h\}$ of $\mathcal{O}(N)$ elements. Assume u is sufficiently smooth and not piecewise polynomials, then we have

$$c(u)n^{-\frac{k}{d}} \leq \inf_{v_h\in V_h^k} \|u - v_h\|_{L^2(\Omega)} \leq C(u)n^{-\frac{k}{d}} = \mathcal{O}(h^k). \tag{26}$$

Ref: Q. Lin, H. Xie and J. Xu , Lower Bounds of the Discretization Error for Piecewise Polynomials, Math. Comp., 83, 1-13 (2014)

[10] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Removing[12] the constraint that $\sum_{i=1}^n |a_i| \le M$

Define

$$\Sigma_n^k := \left\{ \sum_{i=1}^n a_i \sigma_k(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \right\} \tag{27}$$

Theorem (Siegel and Xu)

$$\inf_{f_n \in \Sigma_n^k} \|f - f_n\|_\Omega \lesssim \begin{cases} n^{-\frac{1}{2}} & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \text{if} \quad s = \frac{1}{2} \\ n^{-(k+1)} \log n & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \text{for some} \quad s > 1 \end{cases} \tag{28}$$

- Improves result of Barron[11] by relaxing condition on f

[11] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

[12] Jonathan W Siegel and Jinchao Xu. "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions". In: *arXiv preprint arXiv:2012.07205* (2020).

Removing[12] the constraint that $\sum_{i=1}^{n} |a_i| \leq M$

Define

$$\Sigma_n^k := \left\{ \sum_{i=1}^{n} a_i \sigma_k(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \right\} \tag{27}$$

Theorem (Siegel and Xu)

$$\inf_{f_n \in \Sigma_n^k} \|f - f_n\|_\Omega \lesssim \begin{cases} n^{-\frac{1}{2}} & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \text{if} \quad s = \frac{1}{2} \\ n^{-(k+1)} \log n & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \text{for some} \quad s > 1 \end{cases} \tag{28}$$

- Improves result of Barron[11] by relaxing condition on f
- Shows that very high order approximation rates can be attained with sufficient smoothness

[11] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

[12] Jonathan W Siegel and Jinchao Xu. "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions". In: *arXiv preprint arXiv:2012.07205* (2020).

Removing[12] the constraint that $\sum_{i=1}^n |a_i| \le M$

Define

$$\Sigma_n^k := \left\{ \sum_{i=1}^n a_i \sigma_k(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \right\} \tag{27}$$

Theorem (Siegel and Xu)

$$\inf_{f_n \in \Sigma_n^k} \|f - f_n\|_\Omega \lesssim \begin{cases} n^{-\frac{1}{2}} & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \text{if} \quad s = \frac{1}{2} \\ n^{-(k+1)} \log n & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \text{for some} \quad s > 1 \end{cases} \tag{28}$$

- Improves result of Barron[11] by relaxing condition on f
- Shows that very high order approximation rates can be attained with sufficient smoothness
- Comparison with FEM:

$$\inf_{w \in V_n^k(NN)} \|u - w\| \approx \left\{ \inf_{v \in V_n^k(FE)} \|u - v\| \right\}^d .$$

[11] Andrew R Barron. “Universal approximation bounds for superpositions of a sigmoidal function”. In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

[12] Jonathan W Siegel and Jinchao Xu. “High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions”. In: *arXiv preprint arXiv:2012.07205* (2020).

Model problem

(for any $d \geq 1$, $m \geq 1$)

Given $\Omega \subset \mathbb{R}^d$, consider a $2m$-th order elliptic problems

$$\sum_{|\alpha|=m} (-1)^m \partial^\alpha (a_\alpha(x)\, \partial^\alpha u) + u = f \quad \text{in } \Omega.$$

Model problem

(for any $d \geq 1$, $m \geq 1$)

Given $\Omega \subset \mathbb{R}^d$, consider a $2m$-th order elliptic problems

$$\sum_{|\alpha|=m} (-1)^m \partial^\alpha (a_\alpha(x)\, \partial^\alpha u) + u = f \quad \text{in } \Omega.$$

Special cases:

Model problem

(for any $d \geq 1$, $m \geq 1$)

Given $\Omega \subset \mathbb{R}^d$, consider a $2m$-th order elliptic problems

$$\sum_{|\alpha|=m} (-1)^m \partial^\alpha (a_\alpha(x)\, \partial^\alpha u) + u = f \quad \text{in } \Omega.$$

Special cases:

$$-\Delta u = f \quad (m = 1),$$

Model problem

(for any $d \geq 1$, $m \geq 1$)

Given $\Omega \subset \mathbb{R}^d$, consider a $2m$-th order elliptic problems

$$\sum_{|\alpha|=m} (-1)^m \partial^\alpha (a_\alpha(x)\, \partial^\alpha\, u) + u = f \quad \text{in } \Omega.$$

Special cases:

$$-\Delta u = f \quad (m = 1), \quad \Delta^2 u = f \quad (m = 2).$$

Open Problem: For any $m, d \geq 1$, how to construct conforming finite element space

$$V_h \subset H^m(\Omega) \Longleftrightarrow V_h \subset C^{m-1}(\Omega)?$$

Nonconforming finite element method

Nonconforming finite element method

- Variational "crime":

$$V_h \not\subset V$$

Nonconforming finite element method

- Variational "crime":

$$V_h \not\subseteq V$$

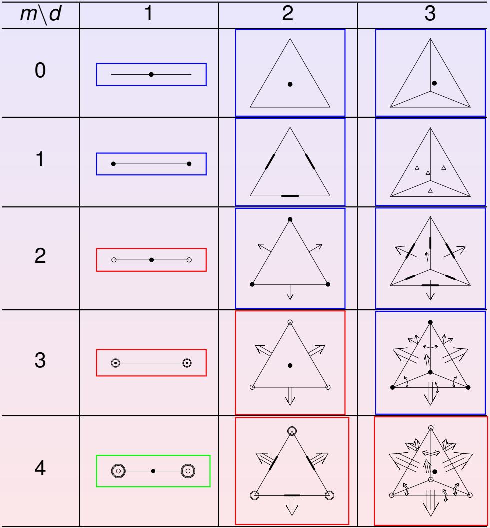
- Bilinear form (with piecewise derivatives: $\partial_h^\alpha v_h$)

$$a_h(u_h, v_h) := \sum_{|\alpha|=m} \sum_{K \in \mathcal{T}_h} (a_\alpha \partial^\alpha u_h, \partial^\alpha v_h)_{0,K} + (u_h, v_h).$$

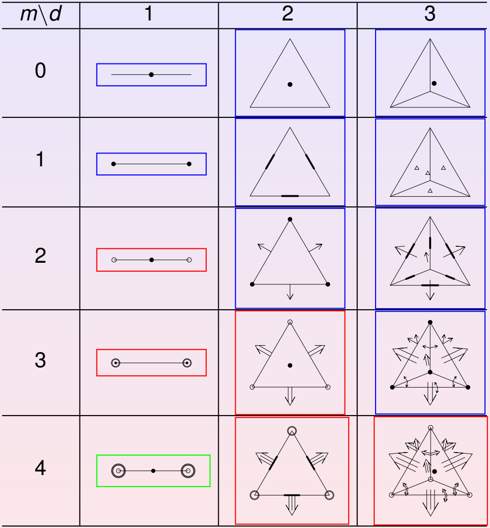
Nonconforming finite element method

- Variational “crime”:
$$V_h \not\subseteq V$$
- Bilinear form (with piecewise derivatives: $\partial_h^\alpha v_h$)
$$a_h(u_h, v_h) := \sum_{|\alpha|=m} \sum_{K \in \mathcal{T}_h} (a_\alpha \partial^\alpha u_h, \partial^\alpha v_h)_{0,K} + (u_h, v_h).$$
- Find $u_h \in V_h$ such that
$$a_h(u_h, v_h) = (f, v_h) \quad \forall v_h \in V_h.$$

Lowest order P_m nonconforming and DG with minimal stabilization



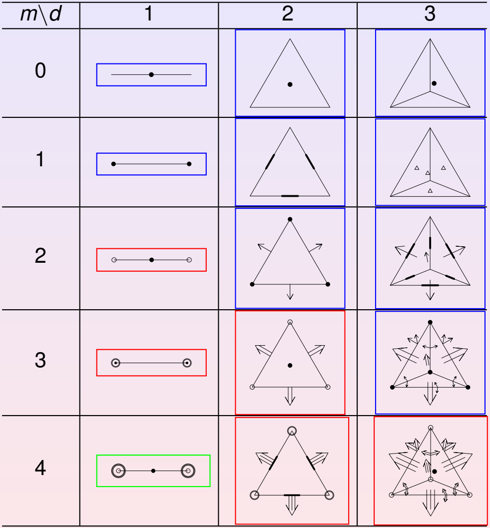
References:



References:

1. $m \leq d$: **Wang, and Xu**, 2013
   - *Minimal finite element spaces for 2m-th-order partial differential equations in $\mathcal{R}^n$*, Mathematics of Computation. 82, 25-43, 2013.

Lowest order P_m nonconforming and DG with minimal stabilization

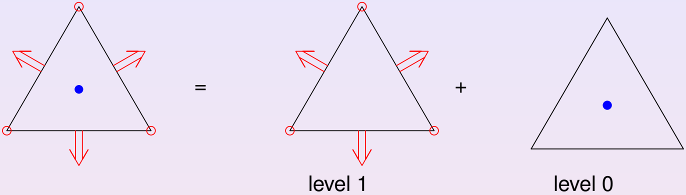


References:

1. $m \leq d$: **Wang, and Xu**, 2013
   - *Minimal finite element spaces for 2m-th-order partial differential equations in $\mathcal{R}^n$*, Mathematics of Computation. 82, 25-43, 2013.
2. $m > d$: **Wu, and Xu** 2017-2020
   - *$\mathcal{P}_m$ interior penalty nonconforming finite element methods for 2m-th Order PDEs in $\mathbb{R}^n$*, arXiv:1710.07678.

Example: $n = d = 2$, $m = 3$

DOF at different levels:



- The highest level ($l = 1$): preserve the crucial property

$$\int_F [\nabla^{m-1} u] = 0.$$

- NO weak continuity for the point value $\Rightarrow$ interior-element-boundary penalty

$$(\nabla_h^3 u_h, \nabla_h^3 v_h) + \eta \sum_{e\in\mathcal{E}_h} h_e^{-5} \int_e [\![u_h]\!] \cdot [\![v_h]\!] = (f, v_h) \qquad \forall v_h \in V_h.$$

(Arnold 1982)

On the construction of smooth FEM

Question: For any $m, d \geq 1$, how to construct conforming finite element space

$$V_h \subset H^m(\Omega) \Longleftrightarrow V_h \subset C^{m-1}(\Omega)?$$

Refs: Argyris et al., (1968); Bramble & Zlámal, (1970); Zhang (2009); Hu & Zhang (2015); Fu, Guzmán & Neilan (2020).

On the construction of smooth FEM

Question: For any $m, d \geq 1$, how to construct conforming finite element space

$$V_h \subset H^m(\Omega) \Longleftrightarrow V_h \subset C^{m-1}(\Omega)?$$

Refs: Argyris et al., (1968); Bramble & Zlámal, (1970); Zhang (2009); Hu & Zhang (2015); Fu, Guzmán & Neilan (2020).

Answer: mostly open, especially when $m \geq 3, d \geq 3$ until recently (2021)

On the construction of smooth FEM

Question: For any $m, d \geq 1$, how to construct conforming finite element space

$$V_h \subset H^m(\Omega) \Longleftrightarrow V_h \subset C^{m-1}(\Omega)?$$

Refs: Argyris et al., (1968); Bramble & Zlámal, (1970); Zhang (2009); Hu & Zhang (2015); Fu, Guzmán & Neilan (2020).

Answer: mostly open, especially when $m \geq 3, d \geq 3$ until recently (2021)

.

Theorem (Hu, Lin, & Wu 2021, ArXiv: 2103.14924))

For any $d \geq 1, r \geq 0$, a globally C^r finite element of degree $k \geq 2^d r + 1$ can be constructed on any simplicial mesh with locally defined DOF.

Conforming elements by neural network: $V_n^k \subset H^m(\Omega)$

Definition:

$$V_n^k = \left\{ \sum_{i=1}^{n} a_i (w_i x + b_i)_+^k, w_i \in \mathbb{R}^{1\times d}, a_i, b_i \in \mathbb{R}^1 \right\}$$

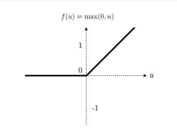
Conforming elements by neural network: $V_n^k \subset H^m(\Omega)$

Definition:

$$V_n^k = \left\{ \sum_{i=1}^{n} a_i (w_i x + b_i)_+^k, w_i \in \mathbb{R}^{1\times d}, a_i, b_i \in \mathbb{R}^1 \right\}$$

where

$$x_+ = \max(0, x) = \mathrm{ReLU}(x)$$



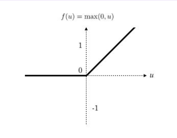
Conforming elements by neural network: $V_n^k \subset H^m(\Omega)$

Definition:

$$V_n^k = \left\{ \sum_{i=1}^n a_i (w_i x + b_i)_+^k, w_i \in \mathbb{R}^{1\times d}, a_i, b_i \in \mathbb{R}^1 \right\}$$

where

$$x_+ = \max(0, x) = \mathrm{ReLU}(x)$$



Properties:

1. Conforming for any $m, d \geq 1$ if $k \geq m$:

$$V_n^k \subset H^k(\Omega) \subset H^m(\Omega)$$

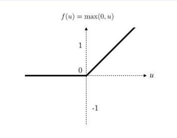
Conforming elements by neural network: $V_n^k \subset H^m(\Omega)$

Definition:

$$V_n^k = \left\{ \sum_{i=1}^{n} a_i (w_i x + b_i)_+^k, w_i \in \mathbb{R}^{1\times d}, a_i, b_i \in \mathbb{R}^1 \right\}$$

where

$$x_+ = \max(0, x) = \mathrm{ReLU}(x)$$

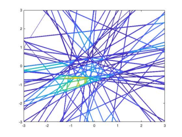


Properties:

1. Conforming for any $m, d \geq 1$ if $k \geq m$:

$$V_n^k \subset H^k(\Omega) \subset H^m(\Omega)$$

2. Piecewise polynomials of degree k in the following grids



Application to high order PDE in any dimension

Consider

$$\begin{cases} Lu = f & \text{in } \Omega, \\ B_N^k(u) = 0, & \text{on } \partial\Omega, \quad 0 \le k \le m-1. \end{cases} \tag{29}$$

Application to high order PDE in any dimension

Consider

$$\begin{cases} Lu = f & \text{in } \Omega, \\ B_N^k(u) = 0, & \text{on } \partial\Omega, \quad 0 \le k \le m-1. \end{cases} \tag{29}$$

$\Longleftrightarrow$ Find $u \in V = H^m(\Omega)$ such that

$$J(u) = \min_{v \in V} J(v) \tag{30}$$

where

$$J(v) = \frac{1}{2} \int_\Omega \sum_{|\alpha|=m} a_\alpha |\partial^\alpha v|^2 + v^2 dx - (f, v). \tag{31}$$

Application to high order PDE in any dimension

Consider

$$\begin{cases} Lu = f & \text{in } \Omega, \\ B_N^k(u) = 0, & \text{on } \partial\Omega, \quad 0 \le k \le m-1. \end{cases} \tag{29}$$

$\Longleftrightarrow$ Find $u \in V = H^m(\Omega)$ such that

$$J(u) = \min_{v \in V} J(v) \tag{30}$$

where

$$J(v) = \frac{1}{2} \int_\Omega \sum_{|\alpha|=m} a_\alpha |\partial^\alpha v|^2 + v^2 dx - (f, v). \tag{31}$$

NN-FEM:

Application to high order PDE in any dimension

Consider

$$\begin{cases} Lu = f & \text{in } \Omega, \\ B_N^k(u) = 0, & \text{on } \partial\Omega, \quad 0 \le k \le m-1. \end{cases} \tag{29}$$

$\Longleftrightarrow$ Find $u \in V = H^m(\Omega)$ such that

$$J(u) = \min_{v \in V} J(v) \tag{30}$$

where

$$J(v) = \frac{1}{2} \int_\Omega \sum_{|\alpha|=m} a_\alpha |\partial^\alpha v|^2 + v^2 dx - (f, v). \tag{31}$$

NN-FEM:Find $u_n \in V_n^k$ as follows:

$$J(u_n) = \min_{v \in V_n^k} J(v). \tag{32}$$

Application to high order PDE in any dimension

Consider

$$\begin{cases} Lu = f & \text{in } \Omega, \\ B_N^k(u) = 0, & \text{on } \partial\Omega, \quad 0 \le k \le m-1. \end{cases} \tag{29}$$

$\Longleftrightarrow$ Find $u \in V = H^m(\Omega)$ such that

$$J(u) = \min_{v \in V} J(v) \tag{30}$$

where

$$J(v) = \frac{1}{2} \int_\Omega \sum_{|\alpha|=m} a_\alpha |\partial^\alpha v|^2 + v^2 dx - (f, v). \tag{31}$$

NN-FEM:Find $u_n \in V_n^k$ as follows:

$$J(u_n) = \min_{v \in V_n^k} J(v). \tag{32}$$

Theorem:

$$\|u - u_n\|_a = \inf_{v_n \in V_n^k} \|u - v_n\|_a$$

Application to high order PDE in any dimension

Consider

$$\begin{cases} Lu = f & \text{in } \Omega, \\ B_N^k(u) = 0, & \text{on } \partial\Omega, \quad 0 \le k \le m-1. \end{cases} \tag{29}$$

$\Longleftrightarrow$ Find $u \in V = H^m(\Omega)$ such that

$$J(u) = \min_{v \in V} J(v) \tag{30}$$

where

$$J(v) = \frac{1}{2} \int_\Omega \sum_{|\alpha|=m} a_\alpha |\partial^\alpha v|^2 + v^2 dx - (f, v). \tag{31}$$

NN-FEM:Find $u_n \in V_n^k$ as follows:

$$J(u_n) = \min_{v \in V_n^k} J(v). \tag{32}$$

Theorem:

$$\|u - u_n\|_a = \inf_{v_n \in V_n^k} \|u - v_n\|_a = \mathcal{O}(n^{m-(k+1)} \log n). \tag{33}$$

Superconvergence (?) property

For $d = 2, m = 1$, consider

$$\Delta^2 u = f.$$

Superconvergence (?) property

For $d = 2, m = 1$, consider

$$\Delta^2 u = f.$$

1. $k = 2$
   - Morley: $\|u - u_n\|_{2,h} = \mathcal{O}(h^{1}) = \mathcal{O}(n^{-\frac{1}{2}}).$

Superconvergence (?) property

For $d = 2, m = 1$, consider

$$\Delta^2 u = f.$$

1. $k = 2$
   - Morley: $\|u - u_n\|_{2,h} = \mathcal{O}(h^1) = \mathcal{O}(n^{-\frac{1}{2}})$.
   - NN-FEM: $\|u - u_n\|_2 = \mathcal{O}(h^2) = \mathcal{O}(n^{-1})$.

Superconvergence (?) property

For $d = 2, m = 1$, consider

$$\Delta^2 u = f.$$

1. $k = 2$
   - Morley: $\|u - u_n\|_{2,h} = \mathcal{O}(h^1) = \mathcal{O}(n^{-\frac{1}{2}})$.
   - NN-FEM: $\|u - u_n\|_2 = \mathcal{O}(h^2) = \mathcal{O}(n^{-1})$.
2. $k = 5$
   - Argyris: $\|u - u_h\|_2 = \mathcal{O}(h^4) = \mathcal{O}(n^{-2})$.

Superconvergence (?) property

For $d = 2, m = 1$, consider

$$\Delta^2 u = f.$$

1. $k = 2$
   - Morley: $\|u - u_n\|_{2,h} = \mathcal{O}(h^{1}) = \mathcal{O}(n^{-\frac{1}{2}})$.
   - NN-FEM: $\|u - u_n\|_2 = \mathcal{O}(h^{2}) = \mathcal{O}(n^{-1})$.
2. $k = 5$
   - Argyris: $\|u - u_h\|_2 = \mathcal{O}(h^{4}) = \mathcal{O}(n^{-2})$.
   - NN-FEM: $\|u - u_n\|_2 = \mathcal{O}(h^{8}) = \mathcal{O}(n^{-4})$.

Properties of $[\mathrm{ReLU}]^k$-DNN_ℓ

Properties of $[\mathrm{ReLU}]^k$-DNN_ℓ

1. Piecewise polynomials on "curved" elements

Properties of $[\text{ReLU}]^k$-DNN_ℓ

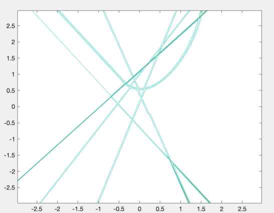
1. Piecewise polynomials on "curved" elements
2. Best possible error estimate $\mathcal{O}(n^{m-(k+1)} \log n)$

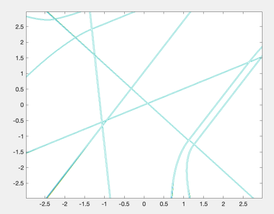
Properties of $[\mathrm{ReLU}]^k$-DNN_ℓ

1. Piecewise polynomials on "curved" elements
2. Best possible error estimate $\mathcal{O}(n^{m-(k+1)}\log n)$
3. If $k \geq 2$, we have spectral accuracy for smooth solution as ℓ increase.

Properties of $[\text{ReLU}]^k$-DNN$_\ell$

1. Piecewise polynomials on "curved" elements
2. Best possible error estimate $\mathcal{O}(n^{m-(k+1)} \log n)$
3. If $k \geq 2$, we have spectral accuracy for smooth solution as ℓ increase.
4. Possible multi-scale adaptivity features (?):
   - local singularity.
   - global smoothness





Some challenges

- Discretization of the integral in $J(u)$, i.e. how do we evaluate

$$\int_{\Omega} |\nabla u(x)|^2 dx - \int_{\Omega} f(x)u(x)dx? \tag{34}$$

Some challenges

- Discretization of the integral in $J(u)$, i.e. how do we evaluate

$$\int_{\Omega} |\nabla u(x)|^2 dx - \int_{\Omega} f(x) u(x) dx? \tag{34}$$

- How to analyze the convergence when numerical quadratures are used?

Some challenges

- Discretization of the integral in $J(u)$, i.e. how do we evaluate

$$\int_{\Omega} |\nabla u(x)|^2 dx - \int_{\Omega} f(x)u(x)dx? \tag{34}$$

- How to analyze the convergence when numerical quadratures are used?
- Optimization of the discrete energy, i.e. how can we efficiently solve

$$\min J_N(u) \tag{35}$$

Discretization of the Integral

There are two approaches for discetizing $J(u)$

- Sample points $x_1, ..., x_N$ uniformly at random from Ω and form

$$J_N(u) = \frac{1}{N}\sum_{i=1}^{N} |\nabla u(x_i)|^2 - f(x_i)u(x_i). \tag{36}$$

Discretization of the Integral

There are two approaches for discetizing $J(u)$

- Sample points $x_1, ..., x_N$ uniformly at random from Ω and form

$$J_N(u) = \frac{1}{N}\sum_{i=1}^{N} |\nabla u(x_i)|^2 - f(x_i)u(x_i). \tag{36}$$

- Use a numerical quadrature rule such as Gaussian quadrature

$$J_N(u) = \sum_{i=1}^{N} a_i(|\nabla u(x_i)|^2 - f(x_i)u(x_i)). \tag{37}$$

Error analysis

Numerical quadrature: for any $g(x)$, $N = \frac{(k-1)d}{2}$

$$\left| \int_\Omega g(x)dx - |\Omega| \sum_{i=1}^{N} w_i g(x_i) \right| \lesssim N^{-\frac{r+1}{d}} \|g\|_{r,1}.$$

Challenges: how to bound

$$\|g\|_{r,1} \le ?, \quad \text{for} \quad g \in \Sigma_n^\sigma$$

OK if the following Bernstein or inverse inequality holds for $r > s$

$$\|v_n\|_r \lesssim n^\beta \|v_n\|_s, \quad \forall v_n \in \Sigma_n^k. \tag{38}$$

Many attempts have been made in existing literature

Bad news: Bernstein inequalty does not hold for NN

Given any $\epsilon > 0$, consider an NN function with 3 neurons:

$$u_3(x) = \mathrm{ReLU}(x - \frac{1}{2} + \epsilon) - 2\mathrm{ReLU}(x - \frac{1}{2}) + \mathrm{ReLU}(x - \frac{1}{2} - \epsilon), \quad \forall x \in (0,1).$$

A direct calculation shows that

$$\int_0^1 |u_3'(x)|^2 dx = 2\epsilon \quad \text{and} \quad \int_0^1 |u_3(x)|^2 dx = \epsilon^2.$$

Therefore

$$|u_3|_{H^1} = \sqrt{\frac{2}{\epsilon}} \|u_3\|_{L^2}, \quad \forall \epsilon > 0$$

As a result, the following Bernstein inequality can not hold for any constant[13] $C(n)$

$$|v_n|_{H^1} \le C(n) \|v_n\|_{L^2}, \quad \forall v_n \in \Sigma_n^\sigma$$

[13] Qingguo Hong, Jonathan W Siegel, and Jinchao Xu. "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs". In: *arXiv preprint arXiv:2104.02903* (2021).

Our approach

Development and analysis of stable neural network!

The use of $\mathcal{K}_1(\mathbb{D})$

- We consider the following variational form of Laplace's equation with Neumann boundary conditions

$$\min_{v \in H^1(\Omega)} J(v) := \int_\Omega |\nabla v(x)|^2 dx - \int_\Omega f(x)v(x)dx. \tag{39}$$

The use of $\mathcal{K}_1(\mathbb{D})$

- We consider the following variational form of Laplace's equation with Neumann boundary conditions

$$\min_{v\in H^1(\Omega)} J(v) := \int_\Omega |\nabla v(x)|^2 dx - \int_\Omega f(x)v(x)dx. \tag{39}$$

- We solve this problem by restricting

$$\min_{\|v\|_{\mathcal{K}_1(\mathbb{D})}\le M} J(v) := \int_\Omega |\nabla v(x)|^2 dx - \int_\Omega f(x)v(x)dx, \tag{40}$$

for some M.

The use of $\mathcal{K}_1(\mathbb{D})$

- We consider the following variational form of Laplace's equation with Neumann boundary conditions

$$\min_{v\in H^1(\Omega)} J(v) := \int_\Omega |\nabla v(x)|^2 dx - \int_\Omega f(x)v(x)dx. \tag{39}$$

- We solve this problem by restricting

$$\min_{\|v\|_{\mathcal{K}_1(\mathbb{D})}\le M} J(v) := \int_\Omega |\nabla v(x)|^2 dx - \int_\Omega f(x)v(x)dx, \tag{40}$$

  for some M.
- With numerical quadrature

$$\min_{\|v\|_{\mathcal{K}_1(\mathbb{D})}\le M} J_N(v) \approx \int_\Omega |\nabla v(x)|^2 dx - \int_\Omega f(x)v(x)dx, \tag{41}$$

  for some M.

Uniform Bound on the Error

- When using numerical quadrature, we require the dictionary $\mathbb{D}$ to satisfy

$$|\mathbb{D}|_{W^{k,\infty}(\Omega)} := \sup_{d\in\mathbb{D}} \|d\|_{W^{k,\infty}(\Omega)} \leq C < \infty. \tag{42}$$

This means that $\|u\|_{W^{k,\infty}(\Omega)} \leq C\|u\|_{\mathcal{K}_1(\mathbb{D})}$.

[14] Qingguo Hong, Jonathan W Siegel, and Jinchao Xu. "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs". In: *arXiv preprint arXiv:2104.02903* (2021).

Uniform Bound on the Error

- When using numerical quadrature, we require the dictionary $\mathbb{D}$ to satisfy

$$|\mathbb{D}|_{W^{k,\infty}(\Omega)} := \sup_{d\in\mathbb{D}} \|d\|_{W^{k,\infty}(\Omega)} \le C < \infty. \tag{42}$$

  This means that $\|u\|_{W^{k,\infty}(\Omega)} \le C\|u\|_{\mathcal{K}_1(\mathbb{D})}$.

- So if we use r-th order quadrature, we will get[14]

$$|J_N(u) - J(u)| \lesssim N^{-\frac{r+1}{d}}, \tag{43}$$

  uniformly on $\{u : \|u\|_{\mathcal{K}_1(\mathbb{D})} \le M\}$.

[14] Qingguo Hong, Jonathan W Siegel, and Jinchao Xu. "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs". In: *arXiv preprint arXiv:2104.02903* (2021).

Uniform Bound on the Error (cont.)

- The Rademacher complexity of a class of function $\mathcal{F}$ on Ω is given by

$$R_N(F) = \mathbb{E}_{x_1,\ldots,x_N}\mathbb{E}_{\xi_1,\ldots,\xi_N}\left(\sup_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} \xi_i f(x_i)\right), \tag{44}$$

where x_i are drawn uniformly at random from Ω and ξ_i are uniformly random signs.

[15] Qingguo Hong, Jonathan W Siegel, and Jinchao Xu. "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs". In: *arXiv preprint arXiv:2104.02903* (2021).

Uniform Bound on the Error (cont.)

- The Rademacher complexity of a class of function $\mathcal{F}$ on Ω is given by

$$R_N(F) = \mathbb{E}_{x_1,\dots,x_N}\mathbb{E}_{\xi_1,\dots,\xi_N}\left(\sup_{f\in\mathcal{F}}\frac{1}{N}\sum_{i=1}^{N}\xi_i f(x_i)\right), \tag{44}$$

  where x_i are drawn uniformly at random from Ω and ξ_i are uniformly random signs.
- For Monte Carlo error analysis, we need to assume that

$$R_N(\mathbb{D}), R_N(\nabla\mathbb{D}) \lesssim N^{-\frac{1}{2}}. \tag{45}$$

- Then we get[15]

$$\mathbb{E}\left(\sup_{\|u\|_{\mathcal{K}_1(\mathbb{D})}\le M}|J_N(u) - J(u)|\right) \lesssim MN^{-\frac{1}{2}}. \tag{46}$$

[15] Qingguo Hong, Jonathan W Siegel, and Jinchao Xu. "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs". In: *arXiv preprint arXiv:2104.02903* (2021).

Orthogonal Greedy Algorithm

The orthogonal greedy algorithm is given by:

- Orthogonal greedy algorithm[16]:

$$f_0 = 0,\ g_k = \arg\max_{g \in \mathbb{D}} \langle f - f_{k-1}, g \rangle,\ f_k = P_k f, \tag{47}$$

where P_k denotes the orthogonal projection onto the space spanned by $g_1, ..., g_k$.

[16] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Orthogonal Greedy Algorithm

The orthogonal greedy algorithm is given by:

- Orthogonal greedy algorithm[16]:

$$f_0 = 0,\ g_k = \arg\max_{g\in\mathbb{D}}\langle f - f_{k-1}, g\rangle,\ f_k = P_k f, \tag{47}$$

  where P_k denotes the orthogonal projection onto the space spanned by $g_1, ..., g_k$.
- There are also the pure greedy and relaxed greedy algorithms

[16] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rates of the Orthogonal Greedy Algorithm

The convergence rates of the orthogonal greedy algorithm is:

[17] Ronald A DeVore and Vladimir N Temlyakov. “Some remarks on greedy algorithms”. In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rates of the Orthogonal Greedy Algorithm

The convergence rates of the orthogonal greedy algorithm is:

- Orthogonal greedy algorithm[17]: $O(n^{-\frac{1}{2}})$

[17] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rates of the Orthogonal Greedy Algorithm

The convergence rates of the orthogonal greedy algorithm is:

- Orthogonal greedy algorithm[17]: $O(n^{-\frac{1}{2}})$
- Similar convergence rates for the pure and relaxed greedy algorithms

[17] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rates of the Orthogonal Greedy Algorithm

The convergence rates of the orthogonal greedy algorithm is:

- Orthogonal greedy algorithm[17]: $O(n^{-\frac{1}{2}})$
- Similar convergence rates for the pure and relaxed greedy algorithms

Can any of these rates be improved for the dictionaries $\mathbb{P}_k^d$ or $\mathbb{F}_s^d$?

[17] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rates of the Orthogonal Greedy Algorithm

The convergence rates of the orthogonal greedy algorithm is:

- Orthogonal greedy algorithm[17]: $O(n^{-\frac{1}{2}})$
- Similar convergence rates for the pure and relaxed greedy algorithms

Can any of these rates be improved for the dictionaries $\mathbb{P}_k^d$ or $\mathbb{F}_s^d$?

- Higher order approximation rates are possible!

[17] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rates of the Orthogonal Greedy Algorithm

The convergence rates of the orthogonal greedy algorithm is:

- Orthogonal greedy algorithm[17]: $O(n^{-\frac{1}{2}})$
- Similar convergence rates for the pure and relaxed greedy algorithms

Can any of these rates be improved for the dictionaries $\mathbb{P}_k^d$ or $\mathbb{F}_s^d$?

- Higher order approximation rates are possible!
- Can the orthogonal greedy algorithms attain them?

[17] Ronald A DeVore and Vladimir N Temlyakov. "Some remarks on greedy algorithms". In: *Advances in computational Mathematics* 5.1 (1996), pp. 173–187.

Convergence Rate of the Orthogonal Greedy Algorithm[18]

Theorem

Let the iterates f_n be given by the orthogonal greedy algorithm, where $f \in \mathcal{K}_1(\mathbb{P}_k^d)$. Then we have

$$\|f_n - f\| \lesssim n^{-\frac{1}{2}-\frac{2k+1}{2d}}. \tag{48}$$

- The orthogonal greedy algorithm can train optimal neural networks!

[18] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Optimization of the Discrete Energy: Greedy Algorithm

We solve the optimization problem

$$\min_{\|u\|_{\mathcal{K}_1(\mathbb{D})}\leq M} J_N(u) \tag{49}$$

using the following greedy algorithm:

$$\begin{aligned} u_0 &= 0 \\ g_k &= \arg\max_{g\in\mathbb{D}} \langle \nabla J_N(u_{k-1}), g\rangle \\ u_k &= (1-s_k)u_{k-1} - Ms_k g. \end{aligned} \tag{50}$$

Optimization of the Discrete Energy: Greedy Algorithm

We solve the optimization problem

$$\min_{\|u\|_{\mathcal{K}_1(\mathbb{D})} \le M} J_N(u) \tag{49}$$

using the following greedy algorithm:

$$\begin{aligned} u_0 &= 0 \\ g_k &= \arg\max_{g\in\mathbb{D}} \langle \nabla J_N(u_{k-1}), g\rangle \\ u_k &= (1 - s_k)u_{k-1} - Ms_k g. \end{aligned} \tag{50}$$

Theorem

$\|u_n\|_{\mathcal{K}_1(\mathbb{D})} \le M$ for all k and

$$J_N(u_n) - \min_{\|u\|_{\mathcal{K}_1(\mathbb{D})} \le M} J_N(u) \lesssim \frac{1}{n}. \tag{51}$$

Main Theorem[19]

Theorem

Suppose that the dictionary $\mathbb{D}$ *satisfies* $\sup_{d\in\mathbb{D}} \|d\|_{W^{1,\infty}(\Omega)} < \infty$ *and the Rademacher complexity bound*

$$R_N(\nabla\mathbb{D}), R_N(\mathbb{D}) \lesssim N^{-\frac{1}{2}}. \quad (52)$$

Assume that the true solution $u \in \mathcal{K}_1(\mathbb{D})$ *satisfies* $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$ *and let the numerical solution* $u_{n,M,N} \in \Sigma_{n,M}(\mathbb{D})$ *be obtained by the greedy algorithm for n steps. Then we have*

$$\mathbb{E}_{x_1,\ldots,x_N}(J(u_{n,M,N}) - J(u)) \leq M\left[C_1(1 + \|f\|_{L^\infty(\Omega)})N^{-\frac{1}{2}} + C_2 M n^{-1}\right]. \quad (53)$$

[19] Qingguo Hong, Jonathan W Siegel, and Jinchao Xu. "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs". In: *arXiv preprint arXiv:2104.02903* (2021).

Summary of the Method

- Need to know M such that the true solution u satisfies $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$

[20] Wenrui Hao et al. “An efficient training algorithm for neural networks and applications in PDEs”. In: *In preparation* (2021).

Summary of the Method

- Need to know M such that the true solution u satisfies $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$
- Choose number of sample points $N = \Theta(M^2\epsilon^{-1})$ and number of iterations $n = \Theta(M^2\epsilon^{-1})$

[20] Wenrui Hao et al. "An efficient training algorithm for neural networks and applications in PDEs". In: *In preparation* (2021).

Summary of the Method

- Need to know M such that the true solution u satisfies $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$
- Choose number of sample points $N = \Theta(M^2\epsilon^{-1})$ and number of iterations $n = \Theta(M^2\epsilon^{-1})$
- Form the discrete energy J_N by randomly sampling points x_i:

$$J_N(u) = \sum_{i=1}^{N} |\nabla u(x_i)|^2 - f(x_i)u(x_i) \tag{54}$$

[20] Wenrui Hao et al. "An efficient training algorithm for neural networks and applications in PDEs". In: *In preparation* (2021).

Summary of the Method

- Need to know M such that the true solution u satisfies $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$
- Choose number of sample points $N = \Theta(M^2\epsilon^{-1})$ and number of iterations $n = \Theta(M^2\epsilon^{-1})$
- Form the discrete energy J_N by randomly sampling points x_i:

$$J_N(u) = \sum_{i=1}^{N} |\nabla u(x_i)|^2 - f(x_i)u(x_i) \tag{54}$$

- Optimize J_N using the relaxed greedy algorithm for n steps

[20] Wenrui Hao et al. "An efficient training algorithm for neural networks and applications in PDEs". In: *In preparation* (2021).

Summary of the Method

- Need to know M such that the true solution u satisfies $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$
- Choose number of sample points $N = \Theta(M^2\epsilon^{-1})$ and number of iterations $n = \Theta(M^2\epsilon^{-1})$
- Form the discrete energy J_N by randomly sampling points x_i:

$$J_N(u) = \sum_{i=1}^{N} |\nabla u(x_i)|^2 - f(x_i)u(x_i) \tag{54}$$

- Optimize J_N using the relaxed greedy algorithm for n steps
- Error will be $O(\epsilon)$

[20] Wenrui Hao et al. “An efficient training algorithm for neural networks and applications in PDEs”. In: *In preparation* (2021).

Summary of the Method

- Need to know M such that the true solution u satisfies $\|u\|_{\mathcal{K}_1(\mathbb{D})} \leq M$
- Choose number of sample points $N = \Theta(M^2\epsilon^{-1})$ and number of iterations $n = \Theta(M^2\epsilon^{-1})$
- Form the discrete energy J_N by randomly sampling points x_i:

$$J_N(u) = \sum_{i=1}^{N} |\nabla u(x_i)|^2 - f(x_i)u(x_i) \tag{54}$$

- Optimize J_N using the relaxed greedy algorithm for n steps
- Error will be $O(\epsilon)$
- Next we will present some numerical experiments[20]

---

[20] Wenrui Hao et al. “An efficient training algorithm for neural networks and applications in PDEs”. In: *In preparation* (2021).

Numerical experiments

Example (2D approximation, OGA)

We consider approximating the following 2D function

$$f(x, y) = \cos(2\pi x)\cos(2\pi y), (x, y) \in (0, 1)^2.$$

By fixing $\|w\| = 1$ and $b \in [-2, 2]$, the convergence order of OGA is shown in Table below for ReLU^k neural networks. Theoretical order is shown in parenthesis.

N	$k = 1$ ($O(n^{-1.25})$)		$k = 2$ ($O(n^{-1.75})$)		$k = 3$ ($O(n^{-2.25})$)	
	L^2-error	order	L^2-error	order	L^2-error	order
2	4.969e-01	-	4.998e-01	-	4.976e-01	-
4	4.883e-01	0.025	4.992e-01	0.002	4.957e-01	0.006
8	2.423e-01	1.011	3.233e-01	0.627	4.193e-01	0.242
16	6.632e-02	1.869	4.911e-02	2.719	1.099e-01	1.932
32	2.206e-02	1.588	1.688e-02	1.541	8.075e-03	3.767
64	1.060e-02	1.058	4.156e-03	2.022	1.149e-03	2.813
128	4.284e-03	1.306	9.773e-04	2.088	2.185e-04	2.395
256	1.703e-03	1.331	2.622e-04	1.898	4.718e-05	2.211

Table: Convergence order of OGA with ReLU^k activation function

Numerical experiments

Example (1D elliptic equation, OGA)

We solve a 1D elliptic equation with the source term $f = \left(1 + \pi^2\right) \cos\left(\pi x\right)$ on $[-1, 1]$ then the analytical solution is $u(x) = \cos\left(\pi x\right), x \in (-1, 1)$. The activation function is ReLU^2.

N	$\\|u - u_N\\|_{L^2}$	order (n^{-3})	$\\|u - u_N\\|_{H^1}$	order (n^{-2})
2	1.312179e+00	-	3.123769e+00	-
4	3.809296e-01	1.78	1.795590e+00	0.80
8	7.900097e-03	5.59	1.239320e-01	3.86
16	6.253874e-04	3.66	2.431156e-02	2.35
32	7.539756e-05	3.05	5.645258e-03	2.11
64	8.098691e-06	3.22	1.351523e-03	2.06
128	9.655067e-07	3.07	3.200813e-04	2.08
256	1.209074e-07	3.00	7.899931e-05	2.02

Table: L^2 and H^1 numerical error of the numerical solution, u_N, where N denotes the number of basis functions.

Numerical experiments

Example (Mesh adaptivity in 1D, OGA)

Let $\Omega = (-1, 1)$ and $K = 0.01$. The solution for 1D elliptic equation is taken with three peaks:

$$u(x) = (1+x)^2(1-x^2)\left(0.5\exp\left(-\frac{(x+0.5)^2}{K}\right) + \exp\left(-\frac{x^2}{K}\right) + 0.5\exp\left(-\frac{(x-0.5)^2}{K}\right)\right).$$

We illustrate the adaptivity by defining the grid points $x = (x_1, \cdots, x_N)^T$ such that $w_1 x + b_1 = 0$.

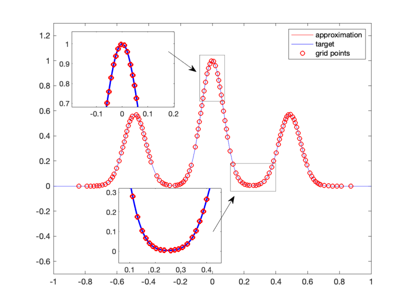


Figure: Grid points of a 1-hidden layer neural network solution with $N = 128$

Numerical experiments

Example (2D 4th order problem, OGA)

Consider the $\|\cdot\|_a$ and $\|\cdot\|_0$ error. We solve this forth-order equation numerically by using the ReLU^3 dictionary

$$\mathbb{D} = \{\mathrm{ReLU}^3(w \cdot x + b) \big| \|w\| = 1, b \in [-2, 2]\}.$$

The exact solution is $(x^2-1)^4(y^2-1)^4$, $(x,y) \in \Omega = (-1,1)^2$.

N	$\|u-u_N\|_{L^2}$	order	$\|u-u_N\|_a$	order ($n^{-1.25}$)
2	6.527642e-01	-	7.926637e+00	-
4	7.859126e-01	-0.27	7.592753e+00	0.06
8	9.906278e-01	-0.33	6.295085e+00	0.27
16	8.215047e-01	0.27	4.002859e+00	0.65
32	1.512860e-01	2.44	1.446132e+00	1.47
64	7.206241e-02	1.07	4.746744e-01	1.61
128	2.258788e-02	1.67	1.808527e-01	1.39
256	4.696294e-03	2.27	6.970084e-02	1.38

Table: The $\|\cdot\|_a$ and $\|\cdot\|_0$ error of the numerical solution

Numerical experiments

Example (A nonlinear 2D example, RGA)

Consider the 2D nonlinear PDE $-\Delta u + u^3 + u = f$ on $(0,1)^2$ with $\partial u/\partial n = 0$ on the boundary. The analytical solution is $u = \cos(2\pi x)\cos(2\pi y)$ and the dictionary is taken as

$$\mathbb{D} = \{\sigma(w_1 x + w_2 y + b) | (w_1, w_2, b) \in [-20, 20]^3\},$$

where $\sigma(x)$ is the sigmoid function. The convergence is considered on the approximating space $B_M(\mathbb{D})$ where $M = 15$.

N	$\|u - u_N\|_2$	order	$\|Du - Du_N\|_2$	order	$J(u_N) - J(u)$	order (n^{-1})
16	7.847118e-01	-	4.645084e+00	-	1.804723e+04	-
32	6.678914e-01	0.23	2.954645e+00	0.65	7.563223e+03	1.25
64	2.370456e-01	1.49	1.675239e+00	0.82	2.327894e+03	1.70
128	1.216064e-01	0.96	1.087479e+00	0.62	9.679782e+02	1.27
256	6.183769e-02	0.98	5.204851e-01	1.06	2.222200e+02	2.12
512	3.796748e-02	0.70	3.610805e-01	0.53	1.066532e+02	1.06
1024	2.687126e-02	0.50	2.110172e-01	0.77	3.661551e+01	1.54
2048	1.072196e-02	1.33	1.431628e-01	0.56	1.663444e+01	1.14

Table: Convergence order of RGA

A new generation of numerical methods?

- Advantages:
  - Highly flexible
  - Works for high-dimensional problems
  - Highly adaptive and parallelizable
  - Rigorous convergence possible using greedy algorithms!
    - For the first time, rigorous results are possible!

A new generation of numerical methods?

- Advantages:
  - Highly flexible
  - Works for high-dimensional problems
  - Highly adaptive and parallelizable
  - Rigorous convergence possible using greedy algorithms!
    - For the first time, rigorous results are possible!
- Disadvantages:
  - Greedy algorithms are currently expensive
  - Much research must still be done!

Summary

- Deep ReLU neural networks contain finite element spaces

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation
  - Optimal approximation rates for ReLU^k neural networks $O(n^{-\frac{1}{2}-\frac{2k+1}{2d}})$

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation
  - Optimal approximation rates for ReLU^k neural networks $O(n^{-\frac{1}{2}-\frac{2k+1}{2d}})$
  - Higher order approximation rates for ReLU^k networks on highly smooth functions (without ℓ^1 coefficient bound)

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation
  - Optimal approximation rates for ReLU^k neural networks $O(n^{-\frac{1}{2}-\frac{2k+1}{2d}})$
  - Higher order approximation rates for ReLU^k networks on highly smooth functions (without ℓ^1 coefficient bound)
- Using neural network to solve $2m$-th order PDEs:

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation
  - Optimal approximation rates for ReLU^k neural networks $O(n^{-\frac{1}{2}-\frac{2k+1}{2d}})$
  - Higher order approximation rates for ReLU^k networks on highly smooth functions (without ℓ^1 coefficient bound)
- Using neural network to solve $2m$-th order PDEs:
  - Bernstein inequality does not work for the neural network space

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation
  - Optimal approximation rates for ReLU^k neural networks $O(n^{-\frac{1}{2}-\frac{2k+1}{2d}})$
  - Higher order approximation rates for ReLU^k networks on highly smooth functions (without ℓ^1 coefficient bound)
- Using neural network to solve $2m$-th order PDEs:
  - Bernstein inequality does not work for the neural network space
  - Convergence analysis for numerical quadrature and Monte Carlo quadrature

Summary

- Deep ReLU neural networks contain finite element spaces
- Approximation property of shallow neural networks:
  - $\mathcal{K}_1(\mathbb{D})$ is the largest space for stable approximation
  - Optimal approximation rates for ReLU^k neural networks $O(n^{-\frac{1}{2}-\frac{2k+1}{2d}})$
  - Higher order approximation rates for ReLU^k networks on highly smooth functions (without ℓ^1 coefficient bound)
- Using neural network to solve $2m$-th order PDEs:
  - Bernstein inequality does not work for the neural network space
  - Convergence analysis for numerical quadrature and Monte Carlo quadrature
  - Use greedy algorithms to solve discrete energy optimization
- Numerical experiments

References

- J. Xu, The Finite Neuron Method and Convergence Analysis, Commun. Comput. Phys., 28, pp. 1707-1745, (2020).
- J. W. Siegel and J. Xu, "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions." arXiv preprint arXiv:2012.07205 (2020).
- J. W. Siegel and J. Xu, "Approximation rates for neural networks with general activation functions." Neural Networks (2020).
- J. W. Siegel and J. Xu, "Optimal Approximation Rates and Metric Entropy of ReLU k and Cosine Networks" arXiv preprint arXiv:2101.12365 (2021)
- Q. Hong J. W. Siegel and J. Xu "A Priori Analysis of Stable Neural Network Solutions to Numerical PDEs" arXiv preprint arXiv:2104.02903 (2021)

Thank you!