Deep Approximation via Deep Learning

Zuowei Shen

Department of Mathematics
National University of Singapore

Outline

Outline

1. Introduction of approximation theory
2. Approximation of functions by composition
3. Power of composition: rate of approximation

A brief introduction

For a given function $f : \mathbb{R}^d \to \mathbb{R}$ and $\epsilon > 0$, approximation is to find a simple function g such that

$$\|f - g\| < \epsilon.$$

A brief introduction

For a given function $f : \mathbb{R}^d \to \mathbb{R}$ and $\epsilon > 0$, approximation is to find a simple function g such that

$$\|f - g\| < \epsilon.$$

Function $g : \mathbb{R}^n \to \mathbb{R}$ can be as simple as $g(x) = a \cdot x$. To make sense of this approximation, we need to find a map $T : \mathbb{R}^d \mapsto \mathbb{R}^n$, such that

$$\|f - g \circ T\| < \epsilon.$$

A brief introduction

For a given function $f : \mathbb{R}^d \to \mathbb{R}$ and $\epsilon > 0$, approximation is to find a simple function g such that

$$\|f - g\| < \epsilon.$$

Function $g : \mathbb{R}^n \to \mathbb{R}$ can be as simple as $g(x) = a \cdot x$. To make sense of this approximation, we need to find a map $T : \mathbb{R}^d \mapsto \mathbb{R}^n$, such that

$$\|f - g \circ T\| < \epsilon.$$

In practice, we only have sample data $\{(x_i, f(x_i))\}_{i=1}^m$ of f, one needs develop algorithms to find T.

A brief introduction

For a given function $f : \mathbb{R}^d \to \mathbb{R}$ and $\epsilon > 0$, approximation is to find a simple function g such that

$$\|f - g\| < \epsilon.$$

Function $g : \mathbb{R}^n \to \mathbb{R}$ can be as simple as $g(x) = a \cdot x$. To make sense of this approximation, we need to find a map $T : \mathbb{R}^d \mapsto \mathbb{R}^n$, such that

$$\|f - g \circ T\| < \epsilon.$$

In practice, we only have sample data $\{(x_i, f(x_i))\}_{i=1}^m$ of f, one needs develop algorithms to find T.

1. Classical approximation: T is independent of f or data, while n depends on ϵ.
2. Learning: T is learned from data and determined by a few parameters. n depends on ϵ.
3. Deep learning: T is fully learned from data with huge number of parameters. T is a composition of many simple maps, and n can be independent of ϵ.

Classical approximation

- Linear approximation: Given a finite fixed set of generators $\{\phi_1, \ldots, \phi_n\}$, e.g. splines, wavelet frames, finite elements or generators in reproducing kernel Hilbert spaces. Define

  $$T = [\phi_1, \phi_2, \ldots, \phi_n]^\top : \mathbb{R}^d \mapsto \mathbb{R}^n \quad \text{and} \quad g(x) = a \cdot x.$$

  The linear approximation is to find $a \in \mathbb{R}^n$ such that

  $$g \circ T = \sum_{i=1}^{n} a_i \phi_i \sim f$$

  It is linear because $f_1 \sim g_1, f_2 \sim g_2 \Rightarrow f_1 + f_2 \sim g_1 + g_2$.

Classical approximation

- Linear approximation: Given a finite fixed set of generators $\{\phi_1, \ldots, \phi_n\}$, e.g. splines, wavelet frames, finite elements or generators in reproducing kernel Hilbert spaces. Define

  $$T = [\phi_1, \phi_2, \ldots, \phi_n]^\top : \mathbb{R}^d \mapsto \mathbb{R}^n \quad \text{and} \quad g(x) = a \cdot x.$$

  The linear approximation is to find $a \in \mathbb{R}^n$ such that

  $$g \circ T = \sum_{i=1}^{n} a_i \phi_i \sim f$$

  It is linear because $f_1 \sim g_1, f_2 \sim g_2 \Rightarrow f_1 + f_2 \sim g_1 + g_2$.
- The best n-term approximation: Given dictionary $\mathcal{D}$ that can have infinitely many generators , e.g. $\mathcal{D} = \{\phi_i\}_{i=1}^{\infty}$ and define

  $$T = [\phi_1, \phi_2, \ldots,]^\top : \mathbb{R}^d \mapsto \in \mathbb{R}^\infty \quad \text{and} \quad g(x) = a \cdot x$$

  The best n-term approximation of f is to find a with n nonzero terms such that $g \circ T \sim f$.is the best approximation among all the n-term choices

  It is nonlinear because $f_1 \sim g_1, f_2 \sim g_2 \not\Rightarrow f_1 + f_2 \sim g_1 + g_2$, as the support of the a_1 and a_2 depends on f_1 and f_2.

Examples

Consider a function space $L_2(\mathbb{R}^d)$, let $\{\phi_i\}_{i=1}^{\infty}$ be an orthonormal basis of $L_2(\mathbb{R}^d)$.

Examples

Consider a function space $L_2(\mathbb{R}^d)$, let $\{\phi_i\}_{i=1}^{\infty}$ be an orthonormal basis of $L_2(\mathbb{R}^d)$.

Linear approximation

For a given n, $T = [\phi_1, \ldots, \phi_n]^\top$ and $g = a \cdot x$ where $a_j = \langle f, \phi_j \rangle$. Denote $\mathcal{H} = \mathrm{span}\{\phi_1, \ldots, \phi_n\} \subseteq L_2(\mathbb{R}^d)$.
Then,

$$g \circ T = \sum_{i=1}^{n} \langle f, \phi_i \rangle \phi_i$$

is the orthogonal projection onto the space $\mathcal{H}$ and is the best approximation of f from the space $\mathcal{H}$.

Examples

Consider a function space $L_2(\mathbb{R}^d)$, let $\{\phi_i\}_{i=1}^{\infty}$ be an orthonormal basis of $L_2(\mathbb{R}^d)$.

Linear approximation

For a given n, $T = [\phi_1, \ldots, \phi_n]^\top$ and $g = a \cdot x$ where $a_j = \langle f, \phi_j \rangle$. Denote $\mathcal{H} = \mathrm{span}\{\phi_1, \ldots, \phi_n\} \subseteq L_2(\mathbb{R}^d)$.

Then,

$$g \circ T = \sum_{i=1}^{n} \langle f, \phi_i \rangle \phi_i$$

is the orthogonal projection onto the space $\mathcal{H}$ and is the best approximation of f from the space $\mathcal{H}$.

$g \circ T$ provides a good approximation of f when the sequence $\{\langle f, \phi_j \rangle\}_{j=1}^{\infty}$ decays fast as $j \to +\infty$.

Examples

Consider a function space $L_2(\mathbb{R}^d)$, let $\{\phi_i\}_{i=1}^{\infty}$ be an orthonormal basis of $L_2(\mathbb{R}^d)$.

Linear approximation
For a given n, $T = [\phi_1, \ldots, \phi_n]^\top$ and $g = a \cdot x$ where $a_j = \langle f, \phi_j \rangle$. Denote $\mathcal{H} = \mathrm{span}\{\phi_1, \ldots, \phi_n\} \subseteq L_2(\mathbb{R}^d)$.
Then,

$$g \circ T = \sum_{i=1}^{n} \langle f, \phi_i \rangle \phi_i$$

is the orthogonal projection onto the space $\mathcal{H}$ and is the best approximation of f from the space $\mathcal{H}$.
$g \circ T$ provides a good approximation of f when the sequence $\{\langle f, \phi_j \rangle\}_{j=1}^{\infty}$ decays fast as $j \to +\infty$.
Therefore,

1. Linear approximation provides a good approximation for smooth functions.
2. **Advantage:** It is a good approximation scheme for d is small, domain is simple, function form is complicated but smooth.
3. **Disadvantage:** It does not do well if d is big and/or domain of f is complex.

Examples

The best n-term approximation

$T = (\phi_j)_{j=1}^{\infty} : \mathbb{R}^d \mapsto \mathbb{R}^{\infty}$ and $g(x) = a \cdot x$ and each a_j is

$$a_j = \begin{cases} \langle f, \phi_j \rangle, & \text{for the largest } n \text{ terms in the sequence } \{|\langle f, \phi_j \rangle|\}_{j=1}^{\infty} \\ 0, & \text{otherwise.} \end{cases}$$

Examples

The best n-term approximation

$T = (\phi_j)_{j=1}^{\infty} : \mathbb{R}^d \mapsto \mathbb{R}^{\infty}$ and $g(x) = a \cdot x$ and each a_j is

$$a_j = \begin{cases} \langle f, \phi_j \rangle, & \text{for the largest } n \text{ terms in the sequence } \{|\langle f, \phi_j \rangle|\}_{j=1}^{\infty} \\ 0, & \text{otherwise.} \end{cases}$$

The approximation of f by $g \circ T$ depends less on the decay of the sequence $\{|\langle f, \phi_j \rangle|\}_{j=1}^{\infty}$. Therefore,

1. the best n-term approximation is better than the linear approximation when f is nonsmooth.
2. It is not a good scheme if d is big and/or domain of f is complex.

Approximation for deep learning

Given data $\{(x_i, f(x_i))\}_{i=1}^m$.

1. The key of deep learning is to construct a T by the given data and chosen g.

Approximation for deep learning

Given data $\{(x_i, f(x_i))\}_{i=1}^m$.

1. The key of deep learning is to construct a T by the given data and chosen g.
2. T can simplify the domain of f through the change of variables while keeping the key features of the domain of f, so that

Approximation for deep learning

Given data $\{(x_i, f(x_i))\}_{i=1}^m$.

1. The key of deep learning is to construct a T by the given data and chosen g.
2. T can simplify the domain of f through the change of variables while keeping the key features of the domain of f, so that
3. It is robust to approximate f by $g \circ T$.

Classical approximation vs deep learning

For both linear and the best n-term approximations, T is fixed. Neither of them suits for approximating f, when f is defined on a complex domain, e.g manifold in a very high dimensional space.

Classical approximation vs deep learning

For both linear and the best n-term approximations, T is fixed. Neither of them suits for approximating f, when f is defined on a complex domain, e.g manifold in a very high dimensional space.

For deep learning, T is constructed by and adapted to the given data. T changes variables and maps domain of f to mach with that of a simple function g. It is normally used to approximate f with complex domain.

Classical approximation vs deep learning

For both linear and the best n-term approximations, T is fixed. Neither of them suits for approximating f, when f is defined on a complex domain, e.g manifold in a very high dimensional space.

For deep learning, T is constructed by and adapted to the given data. T changes variables and maps domain of f to mach with that of a simple function g. It is normally used to approximate f with complex domain.

What is the mathematics behind this?

Settings: construct a measurable map $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ and a simple function g (e.g. $g = a \cdot x$) from data such that the feature of the domain of f can be rearranged by T to match with those of g. This leads to $g \circ T$ provides a good approximation of f.

Outline

Approximation by compositions (with Qianxiao Li and Cheng Tai)

Question 1: For given f and g, is there a measurable $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ such that $f = g \circ T$?

Approximation by compositions (with Qianxiao Li and Cheng Tai)

Question 1: For given f and g, is there a measurable $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ such that $f = g \circ T$?

Answer: Yes! We have proven

Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ *and* $g : \mathbb{R}^n \to \mathbb{R}$ *and assume* $\mathrm{Im}(f) \subseteq \mathrm{Im}(g)$ *and* g *is continuous. Then, there exists a measurable map* $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ *such that*

$$f = g \circ T, \; a.e.$$

- This is an existence proof. T cannot be written out analytically. This leads to the following relaxed question

Approximation by compositions

Question 2: For arbitrarily given $\epsilon > 0$, can one construct a measurable $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ such that $\|f - g \circ T\| \leq \epsilon$?

Approximation by compositions

Question 2: For arbitrarily given $\epsilon > 0$, can one construct a measurable $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ such that $\|f - g \circ T\| \le \epsilon$?

Answer: Yes!

Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$ and assume $\mathrm{Im}(f) \subseteq \mathrm{Im}(g)$. For an arbitrarily given $\epsilon > 0$, a measurable map $T : \mathbb{R}^d \mapsto \mathbb{R}^n$ can be constructed in terms of f and g, such that

$$\|f - g \circ T\| \le \epsilon$$

- While T can be written out in terms of f and g, T can be complex to be constructed when only sample data of f is given. This leads to

Approximation by compositions

Question 3: Can T be a composition of simple maps? That is, can we write $T = T_1 \circ \cdots \circ T_J$, where each T_i, $i = 1, 2, \ldots, J$ is simple, e.g. "perturbation of identity."

Answer: Yes!

Theorem

Denote $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. For an arbitrarily given $\epsilon > 0$, if $\mathrm{Im}(f) \subseteq \mathrm{Im}(g)$, then there exists J simple maps $T_i, i = 1, 2, \ldots, J$ such that $T = T_1 \circ T_2 \ldots \circ T_J : \mathbb{R}^d \mapsto \mathbb{R}^n$ and

$$\|f - g \circ T_1 \circ \cdots \circ T_J\| \leq \epsilon$$

The proof of existence of T_i, $i = 1, 2, \ldots, J$ is constructive. In fact, an algorithm can be devised to carry it out approximately in practice.

Algorithm

Input: Hypothesis spaces: $\mathcal{I}$, $\mathcal{H}$; Loss functions: L, L';
Tolerance: ϵ

Data: $\{x_i, f(x_i)\}_{i=1}^{N}$

Result: A function f_n that approximates a given f

initialization: Set $f_0 = g$, $\mathrm{Im} g \supset \mathrm{Im} f$;

for j *from* 0 *to* $n-1$ **do**

$I_j = \arg\min_{I \in \mathcal{I}} \quad \frac{1}{N} \sum_{i=1}^{N} L(I(x_i), \mathbb{1}_{\{|f_j - f| > \epsilon\}}(x_i))$;

$h_j = \arg\min_{h \in \mathcal{H}} \quad \frac{1}{N} \sum_{i=1}^{N} L'(f(x_i), f_j \circ T_{h,j}(x_i))$

where $T_{h,j}(x) := I_j(x)h(x) + [1 - I_j(x)]x$;

Set $f_{j+1} = f_j \circ T_{h_j,j}$

end

Advantage of Multi-level Composition

- For any given any approximator, this algorithm systematically improve its performance by adding one more layer of composition

Advantage of Multi-level Composition

- For any given any approximator, this algorithm systematically improve its performance by adding one more layer of composition
- The performance improvement can be quantified by

$$D_\epsilon(f, g \circ t) = D_\epsilon(f, g)\left[1 - r\left(1 - \frac{a}{p}\right)\right]$$

a, r, p can be estimated at each stage to see if we can go further

Advantage of Multi-level Composition

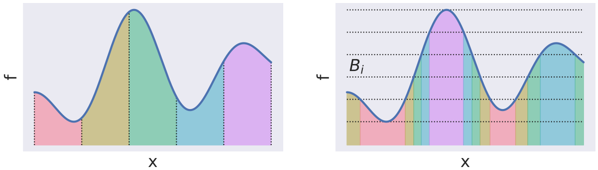
- For any given any approximator, this algorithm systematically improve its performance by adding one more layer of composition
- The performance improvement can be quantified by

$$D_\epsilon(f, g \circ t) = D_\epsilon(f, g) \left[1 - r\left(1 - \frac{a}{p}\right)\right]$$

  a, r, p can be estimated at each stage to see if we can go further
- This procedure also naturally picks up some multi-scale structure

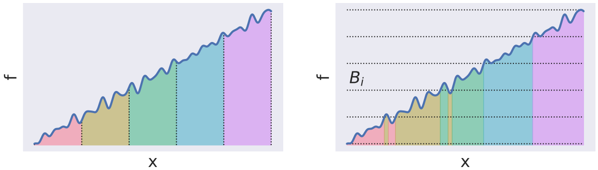
Ideas

- Classical approximation sub-divides the domain, The key to a good approximation is to reproduce poly locally. The smoothness of f is needed. It is a local approach (e.g. Riemann integration, TV method).
- Alternative approach sub-divides the range. The key to good approximation is the location, volume, and geometry of $f^{-1}(B_i)$, The smoothness of f is no more important. It is non-local (e.g. Lebesgue integration, non-local TV method)
- Our theory and algorithm iteratively rearranges $f^{-1}(B_i)$ by constructing T, so that it matches with $g^{-1}(B_i)$, Consequently, $g \circ T$ approximates f well.

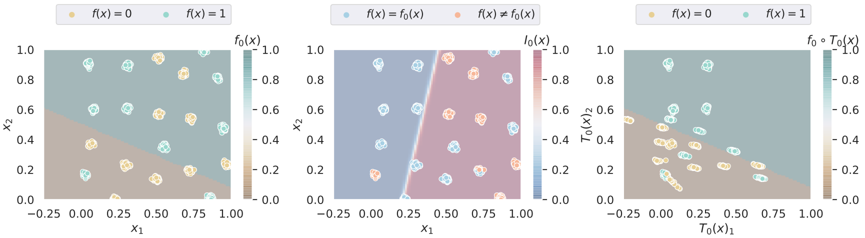


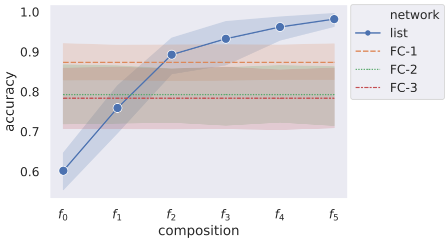
Ideas

- Classical approximation sub-divides the domain, The key to a good approximation is to reproduce poly locally. The smoothness of f is needed. It is a local approach (e.g. Riemann integration, TV method).
- Alternative approach sub-divides the range. The key to good approximation is the location, volume, and geometry of $f^{-1}(B_i)$, The smoothness of f is no more important. It is non-local (e.g. Lebesgue integration, non-local TV method)
- Our theory and algorithm iteratively rearranges $f^{-1}(B_i)$ by constructing T, so that it matches with $g^{-1}(B_i)$, Consequently, $g \circ T$ approximates f well.

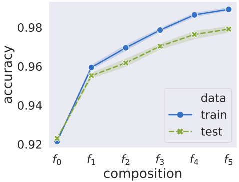


A Binary Classification Toy Problem

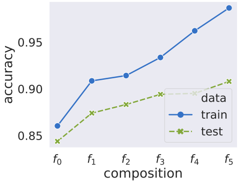




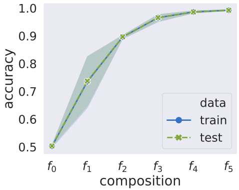
Other Classification and Regression Benchmarks



(a) MNIST



(b) Fashion-MNIST

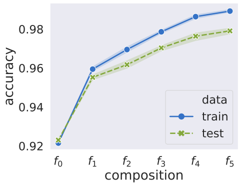


(c) SGEMM[1]

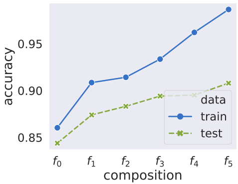
Remark: For the image classification problems, h, I composes of small convolution blocks with 4-32 channels, and 2-4 layers each. f_0 is linear.

[1] Cedric Nugteren and Valeriu Codreanu. MCSoC, 2015 (http://ieeexplore.ieee.org/document/7328205/)

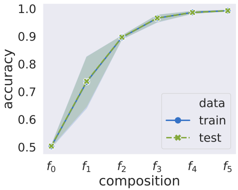
Other Classification and Regression Benchmarks



(d) MNIST



(e) Fashion-MNIST



(f) SGEMM[1]

Remark: The last problem is regression, with fully connected blocks for h, I. "Accuracy" is defined as in the preceding theory: $D_\epsilon(f, f_j) = \mu\{|f - f_j| > \epsilon\}$. Here, we take $\epsilon = 0.1$.

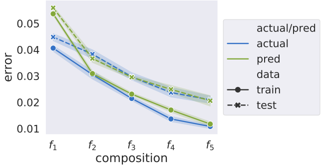
[1] Cedric Nugteren and Valeriu Codreanu. MCSoC, 2015 (http://ieeexplore.ieee.org/document/7328205/)

Validating Theoretical Predictions

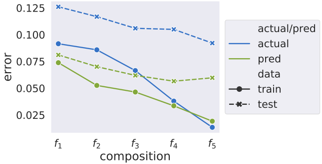
Recall that we can estimate improvement *before* training next layer using $D_\epsilon(f, f_{j+1}) = D_\epsilon(f, f_j)\left[1 - r_j\left(1 - \frac{a_j}{p_j}\right)\right]$ where r, p are precision and recall of the indicator function I_j and a_j is the approximation power of h_j, which can all be estimated/bounded.

Validating Theoretical Predictions

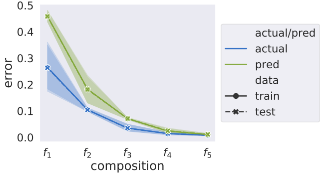
Recall that we can estimate improvement *before* training next layer using $D_\epsilon(f, f_{j+1}) = D_\epsilon(f, f_j)\left[1 - r_j\left(1 - \frac{a_j}{p_j}\right)\right]$



(j) MNIST



(k) Fashion-MNIST



(l) SGEMM

Q. Li, Z. Shen, and C Tai Deep approximation of functions via composition (2019).

j

Outline

The best N-term Approximation via Dictionary with Compositions(with Haizhao Yang and Shijun Zhang)

N-term approximation Given a dictionary $\mathcal{D}$ and f, the best

n-term approximation from $\mathcal{D}$ is to find $\phi_i^* \in \mathcal{D}$ and $a_i^* \in \mathbb{R}$ such that

$$g = \sum_{i=1}^{n} a_i^* \phi_i^*$$

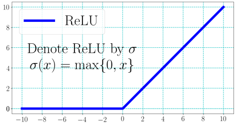
is a solution of

$$\inf_{a_i \in \mathbb{R},\, \phi_i \in \mathcal{D}} \left\| f - \sum_{i=1}^{n} a_i \phi_i \right\|.$$

The best N-term Approximation via Dictionary with Compositions

First dictionary is defined as

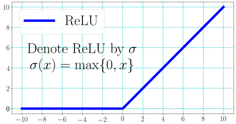
$$\mathcal{D}_1 := \{\sigma(\boldsymbol{W} \cdot \boldsymbol{x} + b) : \boldsymbol{W} \in \mathbb{R}^d,\, b \in \mathbb{R}\}$$



The best N-term Approximation via Dictionary with Compositions

First dictionary is defined as

$$\mathcal{D}_1 := \{\sigma(\boldsymbol{W} \cdot \boldsymbol{x} + b) : \boldsymbol{W} \in \mathbb{R}^d,\, b \in \mathbb{R}\}$$

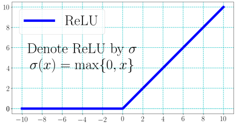


Each element of $\mathcal{D}_1$ is a piecewise linear function.

The best N-term Approximation via Dictionary with Compositions

First dictionary is defined as

$$\mathcal{D}_1 := \{\sigma(\boldsymbol{W} \cdot \boldsymbol{x} + b) : \boldsymbol{W} \in \mathbb{R}^d,\, b \in \mathbb{R}\}$$



Each element of $\mathcal{D}_1$ is a piecewise linear function.

When $d = 1$, for arbitrary Lipchitz continuous f on $[0, 1]$, the best n-term approximation from $\mathcal{D}_1$ achieve the approximation rate $O(n^{-1})$.

The best N-term Approximation via Dictionary with Compositions

Dictionary via compositions:

The best N-term Approximation via Dictionary with Compositions

Dictionary via compositions:

Choosing $h_1, h_2, \cdots, h_n \in \mathcal{D}_1$, denote column vector $[h_1, h_2, \cdots, h_n]^T$ by $\boldsymbol{h}$, the second dictionary is defined as

$$\mathcal{D}_2 := \{\sigma(\boldsymbol{W} \cdot \boldsymbol{h} + b) : \boldsymbol{W} \in \mathbb{R}^n,\ b \in \mathbb{R}\}.$$

The best N-term Approximation via Dictionary with Compositions

Dictionary via compositions:

Choosing $h_1, h_2, \cdots, h_n \in \mathcal{D}_1$, denote column vector $[h_1, h_2, \cdots, h_n]^T$ by $\boldsymbol{h}$, the second dictionary is defined as

$$\mathcal{D}_2 := \{\sigma(\boldsymbol{W} \cdot \boldsymbol{h} + b) : \boldsymbol{W} \in \mathbb{R}^n,\, b \in \mathbb{R}\}.$$

Each element of $\mathcal{D}_2$ is compositions of piecewise linear functions.

The best N-term Approximation via Dictionary with Compositions

Dictionary via compositions:

Choosing $h_1, h_2, \cdots, h_n \in \mathcal{D}_1$, denote column vector $[h_1, h_2, \cdots, h_n]^T$ by $\boldsymbol{h}$, the second dictionary is defined as

$$\mathcal{D}_2 := \{\sigma(\boldsymbol{W} \cdot \boldsymbol{h} + b) : \boldsymbol{W} \in \mathbb{R}^n, \, b \in \mathbb{R}\}.$$

Each element of $\mathcal{D}_2$ is compositions of piecewise linear functions.

Compositions of piecewise linear functions are still piecewise linear functions.

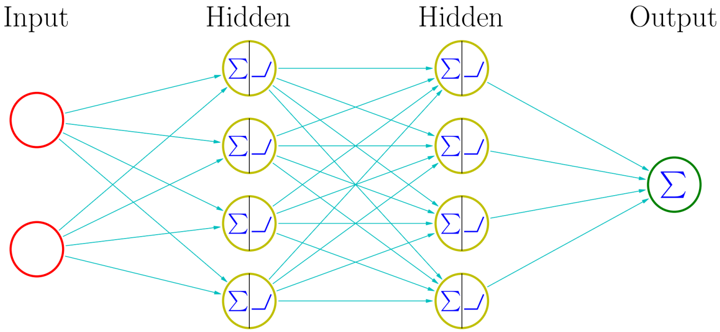
This process can continue inductively to derive multilayer composition dictionaries $\mathcal{D}_3, \ldots \mathcal{D}_L$.

The best N-term Approximation via Dictionary with Compositions

The N-term approximation from $\mathcal{D}_2$ can be implemented numerically by the ReLU networks with 2 hidden layer approximation.

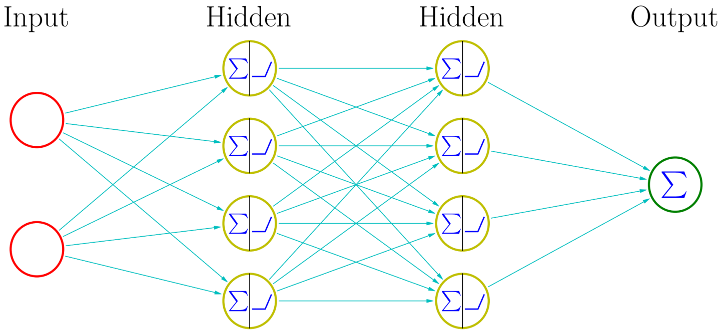
The best N-term Approximation via Dictionary with Compositions

The N-term approximation from $\mathcal{D}_2$ can be implemented numerically by the ReLU networks with 2 hidden layer approximation.



The best N-term Approximation via Dictionary with Compositions

The N-term approximation from $\mathcal{D}_2$ can be implemented numerically by the ReLU networks with 2 hidden layer approximation.



When $d = 1$, for any Lipchitz continuous f on $[0, 1]$, the best n-term approximation from $\mathcal{D}_2$ achieve the approximation rate $O(n^{-2})$.

The best N-term Approximation via Dictionary with Compositions

dictionary	corresponding network	approximation rate
$\mathcal{D}_1$	1 hidden layer	$O(n^{-1})$
$\mathcal{D}_2$	2 hidden layer	$O(n^{-2})$

The Dictionary with composition improves n-term approximation rate!

The best N-term Approximation via Dictionary with Compositions

dictionary	corresponding network	approximation rate
$\mathcal{D}_1$	1 hidden layer	$O(n^{-1})$
$\mathcal{D}_2$	2 hidden layer	$O(n^{-2})$

The Dictionary with composition improves n-term approximation rate!

The best N-term Approximation via Dictionary with Compositions

dictionary	corresponding network	approximation rate
$\mathcal{D}_1$	1 hidden layer	$O(n^{-1})$
$\mathcal{D}_2$	2 hidden layer	$O(n^{-2})$

The Dictionary with composition improves n-term approximation rate!

The best N-term Approximation via Dictionary with Compositions

dictionary	corresponding network	approximation rate
$\mathcal{D}_1$	1 hidden layer	$O(n^{-1})$
$\mathcal{D}_2$	2 hidden layer	$O(n^{-2})$

The Dictionary with composition improves n-term approximation rate!

For any fixed L, can the dictionary $\mathcal{D}_L$ attain the n-term of approximation rate $O(n^{-L})$ for $L \geq 3$?

The answer is unlikely, unless one do something else...

The answer is unlikely, unless one do something else...

The answer is unlikely, unless one do something else...

Given $L \geq 1$, there exists f with Lipchitz constant 1 such that the n-term approximation error from $\mathcal{D}_L$ cannot be better than

$$O(n^{-(2+\rho)})$$

for sufficiently large n and any $\rho > 0$.

The answer is unlikely, unless one do something else...

Given $L \geq 1$, there exists f with Lipchitz constant 1 such that the n-term approximation error from $\mathcal{D}_L$ cannot be better than

$$O(n^{-(2+\rho)})$$

for sufficiently large n and any $\rho > 0$.

Multilayer implying multiplication of the approximate rate is only true for 2 hidden layers but not for $L \geq 3$.

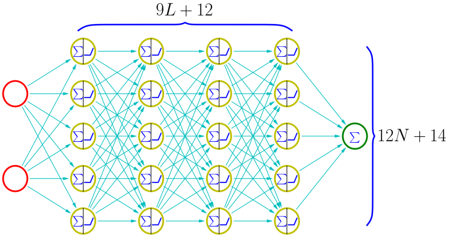
That means that one cannot expect to reach the n -term approximation rate $O(n^{-L})$ for multilayer composition dictionary $\mathcal{D}_L$ for fixed $L \geq 3$.

Z. Shen, H. Yang, and S. Zhang, Nonlinear Approximation via Compositions, arXiv e-prints, (2019), arXiv:1902.10170,601p. arXiv:1902.1017.

Approximation Rate of ReLU Networks

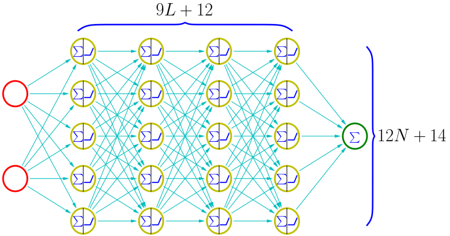
For given $N, L > 1 \in \mathbb{N}^+$, design a network of order $O(NL)$

ϕ=



Approximation Rate of ReLU Networks

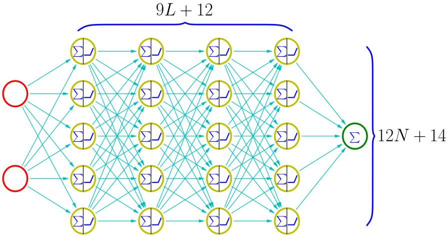
For given $N, L > 1 \in \mathbb{N}^+$, design a network of order $O(NL)$

ϕ= 

Question: What is the approximation rate for this ReLU network?

Approximation Rate of ReLU Networks

For given $N, L > 1 \in \mathbb{N}^+$, design a network of order $O(NL)$

ϕ= 

Question: What is the approximation rate for this ReLU network?

Suppose f is Lipchitz with constant ν, then

$$\|f-\phi\|_{L^p([0,1]^d)} \leq 40\nu\sqrt{d}N^{-2/d}L^{-2/d},$$

for $p \in [1, \infty)$.
When $d > 1$, the width is $\max\{8d\lfloor N^{1/d}\rfloor + 4d, 12N+14\}$.

Approximation Rate of ReLU Networks

For general continuous functions, define the modulus of continuity, for any $r > 0$, as

$$\omega_f(r) := \sup\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d,\ |\boldsymbol{x} - \boldsymbol{y}| \leq r\}.$$

Approximation Rate of ReLU Networks

For general continuous functions, define the modulus of continuity, for any $r > 0$, as

$$\omega_f(r) := \sup\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d,\ |\boldsymbol{x} - \boldsymbol{y}| \le r\}.$$

Theorem

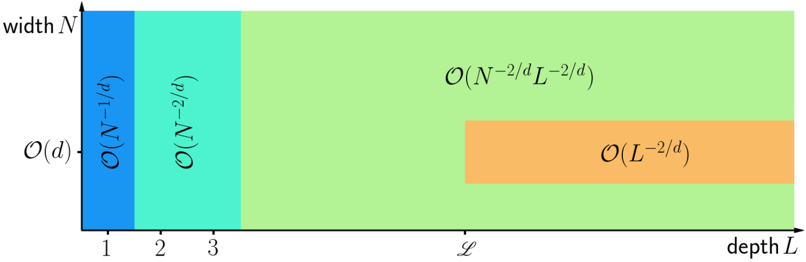
Suppose f is continuous, $\forall\, L > 1, N \in \mathbb{N}^+$ and $\forall\, p \in [1, \infty)$, $\exists$ a ReLU network ϕ with width $\max\left\{8d\lfloor N^{1/d}\rfloor + 4d, 12N + 14\right\}$ and depth $9L + 12$ such that

$$\|f - \phi\|_{L^p([0,1]^d)} \le 5\omega_f(8\sqrt{d}N^{-2/d}L^{-2/d}).$$

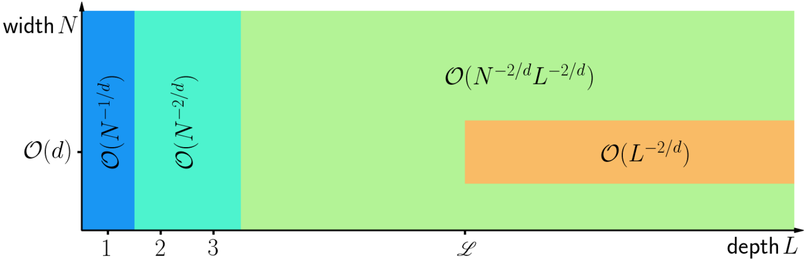
The rate $O(N^{-2/d}L^{-2/d})$ is nearly optimal.

Zuowei Shen, Haizhao Yang, Shijun Zhang. Approximation Rate of Deep ReLU Networks.

Approximation Rate of ReLU Networks

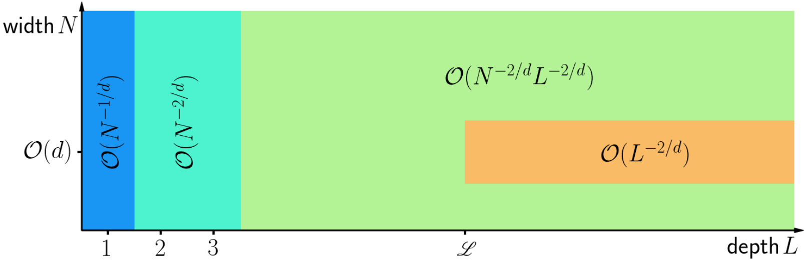


Approximation Rate of ReLU Networks



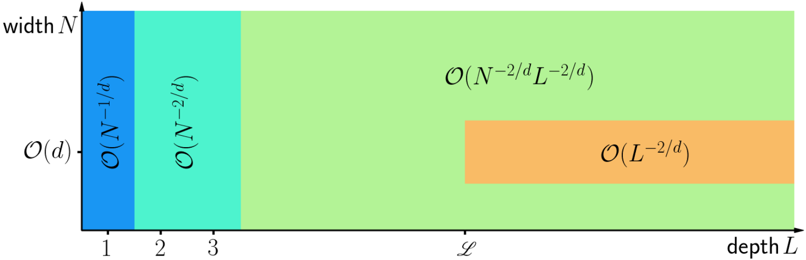
- : $N \geq 1$, $L = 1$, rate $O(N^{-1/d})$, well known.

Approximation Rate of ReLU Networks



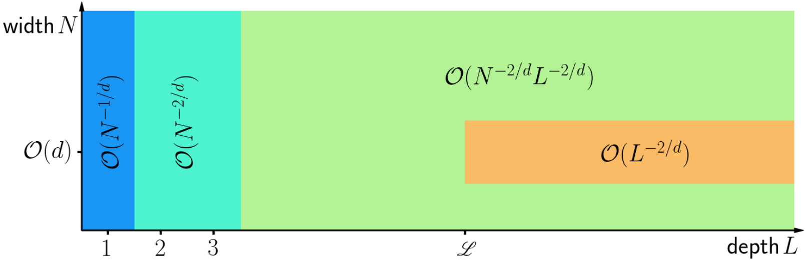
- (blue box) : $N \geq 1$, $L = 1$, rate $O(N^{-1/d})$, well known.
- (orange box) : $N = 2d + 10$, L sufficient large, rate $O(L^{-2/d})$, Yarotsky, 2018.

Approximation Rate of ReLU Networks



- [blue box]: $N \geq 1$, $L = 1$, rate $O(N^{-1/d})$, well known.
- [orange box]: $N = 2d + 10$, L sufficient large, rate $O(L^{-2/d})$, Yarotsky, 2018.
- [turquoise box]: $N \geq 1$, $L = 2, 3$, rate $O(N^{-2/d})$, implied by the results of N-term approximation, Shen, Yang, Zhang, 2019.

Approximation Rate of ReLU Networks



- (blue) : $N \geq 1$, $L = 1$, rate $O(N^{-1/d})$, well known.
- (orange) : $N = 2d + 10$, L sufficient large, rate $O(L^{-2/d})$, Yarotsky, 2018.
- (cyan) : $N \geq 1$, $L = 2, 3$, rate $O(N^{-2/d})$, implied by the results of N-term approximation, Shen, Yang, Zhang, 2019.
- (green) : $N \geq 1$, $L \geq 1$, rate $O(N^{-2/d}L^{-2/d})$, Shen, Yang, Zhang, 2019.

Thank you!

http://www.math.nus.edu.sg/∼matzuows/