Mathematics of Deep Learning, Summer Term 2020

Week 8

ReLU Networks and the Role of Depth

Philipp Harms Lars Niemann

University of Freiburg

Overview of Week 8

Acknowledgement of Sources

Sources for this lecture:

- Philipp Christian Petersen (Faculty of Mathematics, University of Vienna): Course on Neural Network Theory.
- Perekrestenko, Grohs, Elbrächter, Bölcskei (2018): The universal approximation power of finite-width deep ReLU Networks. `arXiv:1806.01528`
- E, Wang (2018): Exponential convergence of the deep neural approximation for analytic functions. `arXiv:1807.00297`
- Yarotsky (2017): Error bounds for approximations with deep ReLU networks. Neural Networks 94, pp. 103–114.

Mathematics of Deep Learning, Summer Term 2020

Week 8, Video 1

Operations on ReLU Networks

Philipp Harms   Lars Niemann

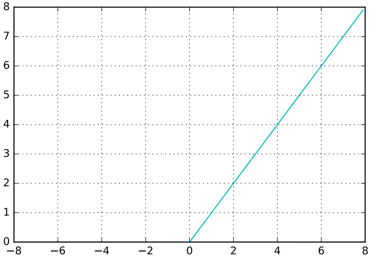
University of Freiburg

Repetition: ReLU Activation Function

Definition

The rectified linear unit (ReLU) activation function is defined as

$$\rho_R(x) = \max(0, x), \qquad x \in \mathbb{R}.$$



Remark: The ReLU function is not sigmoidal but discriminatory.

Networks of Bounded Width with Bounded Weights

Remark:

- Previously, the focus was on wide networks of bounded depth.
- For ReLU networks, we focus on deep networks of bounded width.

Definition

Let $\Phi = ((A_1, b_1), \ldots, (A_L, b_L))$ be a neural network with architecture $(N_0, N_1, \ldots, N_L)$.

- The width of Φ is defined as $\mathrm{W}(\Phi) := \max_i N_i$.
- The weight bound of Φ is defined as

$$\mathrm{B}(\Phi) := \max\{\max_i \|A_i\|_{\infty,\infty}, \max_i \|b_i\|_\infty\},$$

where the norms $\|\cdot\|_{\infty,\infty}$ and $\|\cdot\|_\infty$ are the maxima of the absolute values of the matrix or vector entries, respectively.

ReLU Representation of the Identity

Lemma

For each $d \in \mathbb{N}$ and $L \in \mathbb{N}$, the identity on $\mathbb{R}^d$ can be realized as $\mathrm{Id}_{\mathbb{R}^d} = \mathrm{R}(\Phi^{\mathrm{Id}}_{d,L})$ for a ReLU network $\Phi^{\mathrm{Id}}_{d,L}$ with $\mathrm{B}(\Phi^{\mathrm{Id}}_{d,L}) = 1$, $\mathrm{W}(\Phi^{\mathrm{Id}}_{d,L}) = 2d$, and $\mathrm{L}(\Phi^{\mathrm{Id}}_{d,L}) = L$.

Proof: For $L = 1$ we use $\Phi^{\mathrm{Id}}_{d,1} := ((\mathrm{Id}_{\mathbb{R}^d}, 0))$, and for $L \geq 2$, the network

$$\Phi^{\mathrm{Id}}_{d,L} := \left(\left(\begin{pmatrix} \mathrm{Id}_{\mathbb{R}^d} \\ -\mathrm{Id}_{\mathbb{R}^d} \end{pmatrix}, 0 \right), (\mathrm{Id}_{\mathbb{R}^{2d}}, 0), \ldots (\mathrm{Id}_{\mathbb{R}^{2d}}, 0), ((\mathrm{Id}_{\mathbb{R}^d}, -\mathrm{Id}_{\mathbb{R}^d}), 0) \right)$$

has the desired properties thanks to the algebraic relations

$$\rho_R(x) - \rho_R(-x) = x, \qquad \rho_R(\rho_R(x)) = \rho_R(x). \qquad \square$$

Problem: Lack of Sparsity in Network Concatenations

Example: Lack of sparsity in network concatenations.

- Let $n \in \mathbb{N}$ and define the neural network Φ by

$$\Phi := ((A_1, 0), (A_2, 0)),$$

  where $A_1 = (1, \ldots, 1)^\top \in \mathbb{R}^{n \times 1}$ and $A_2 = (1, \ldots, 1) \in \mathbb{R}^{1 \times n}$.

- Φ realizes the map

$$\mathbb{R} \ni x \mapsto (x, \ldots, x) \mapsto (x_+, \ldots, x_+) \mapsto x_+ + \cdots + x_+ = n x_+ \in \mathbb{R}.$$

- Then $\mathrm{M}(\Phi) = 2n$ but $\mathrm{M}(\Phi \bullet \Phi) = 2n + n^2$ because

$$\Phi \bullet \Phi = ((A_1, 0), (A_1 A_2, 0), (A_2, 0)).$$

- Hence, the number of weights of a concatenated network scales quadratically in the number of weights of the individual networks.

Solution: Sparse Concatenation

Remark: The lack of sparsity of concatenations motivates the following definition:

Definition

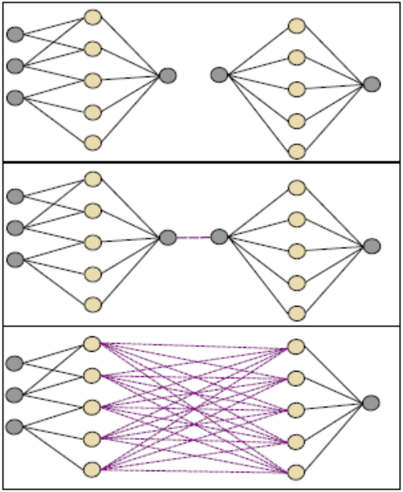
The sparse concatenation of a neural network Φ^1 with input dimension d and neural network Φ_2 with output dimension d is defined as

$$\Phi^1 \odot \Phi^2 := \Phi^1 \bullet \Phi^{\mathrm{Id}}_{d,2} \bullet \Phi^2,$$

where $\Phi^{\mathrm{Id}}_{d,2}$ is the 2-layer ReLU representation of the identity on $\mathbb{R}^d$.

Remark: Similarly, using $\Phi^{\mathrm{Id}}_{d,L}$ with $L > 2$, one can define sparse concatenations of increased depth.

Concatenation versus Sparse Concatenation



Top: Two neural networks, Middle: Sparse Concatenation, Bottom: Concatenation. [Figure from Petersen, Ch. 3]

Properties of Sparse Concatenation

Lemma

If Φ^1 *has input dimension* d *and* Φ_2 *has output dimension* d*, then the* *sparse concatenation* $\Phi^1 \odot \Phi^2$ *satisfies*

$$
\begin{aligned}
\mathrm{R}(\Phi^1 \odot \Phi^2) &= \mathrm{R}(\Phi^1) \circ \mathrm{R}(\Phi^2), \\
\mathrm{L}(\Phi^1 \odot \Phi^2) &= \mathrm{L}(\Phi^1) + \mathrm{L}(\Phi^2), \\
\mathrm{M}(\Phi^1 \odot \Phi^2) &\leq 2(\mathrm{M}(\Phi^1) + \mathrm{M}(\Phi^2)), \\
\mathrm{W}(\Phi^1 \odot \Phi^2) &\leq \max(\mathrm{W}(\Phi^1), \mathrm{W}(\Phi^2), 2d), \\
\mathrm{B}(\Phi^1 \odot \Phi^2) &\leq \max(\mathrm{B}(\Phi^1), \mathrm{B}(\Phi^2)).
\end{aligned}
$$

Remark: Most importantly, the number of weights increases linearly rather than quadratically, and the weights remain bounded.

Proof: Properties of Sparse Concatenation

Proof:

- Sparse concatenation realizes function composition because

$$\mathrm{R}(\Phi^1 \bullet \Phi^{\mathrm{Id}}_{d,2} \bullet \Phi^2) = \mathrm{R}(\Phi^1) \circ \mathrm{R}(\Phi^{\mathrm{Id}}_{d,2}) \circ \mathrm{R}(\Phi^2) = \mathrm{R}(\Phi^1) \circ \mathrm{R}(\Phi^2).$$

- The width, depth, weight bound, and number of weights can be estimated from the following explicit formula:

$$\begin{aligned}
&((A_1^1, b_1^1), \ldots, (A_{L_1}^1, b_{L_1}^1)) \odot ((A_1^2, b_1^2), \ldots, (A_{L_2}^2, b_{L_2}^1)) \\
&\quad = \Bigg((A_1^2, b_1^2), \ldots, (A_{L_2-1}^2, b_{L_2-1}^2), \left(\begin{pmatrix} A_{L_2}^2 \\ -A_{L_2}^2 \end{pmatrix}, \begin{pmatrix} b_{L_2}^2 \\ -b_{L_2}^2 \end{pmatrix} \right), \\
&\qquad\qquad ((A_1^1, -A_1^1), b_1^1), (A_2^1, b_2^1), \ldots, (A_{L_1}^1, b_{L_1}^1) \Bigg).
\end{aligned}$$

□

Skip Connections

Remark: Recall that a network $\Phi = ((A_1, b_1), \ldots, (A_L, b_L))$ can be represented as a computational graph with edges corresponding to the non-zero entries of the matrices A_i.

Definition

A skip connection is an edge between non-adjacent layers in the computational graph of a network.

Remark:

- Networks with skip connections have been highly successful in image recognition.
- The ReLU representation of the identity allows one to rewrite networks with skip connections as networks without skip connections.

Deep Linear Combinations of Networks

Remark:

- The following implementation of linear combinations increases the depth, and not the width, of the networks.
- As scalar multiplication does not affect the network structure, we focus on sums of networks.

Lemma

For any networks $\Phi_1, \ldots, \Phi_k$ *with input dimension* d *and output dimension* n*, there exists a network* Φ *with* $\mathrm{B}(\Phi) \leq \max_i \mathrm{B}(\Phi_i)$*,* $\mathrm{W}(\Phi) \leq \max_i \mathrm{W}(\Phi) + 2d + 2n$*, and* $\mathrm{L}(\Phi) = \sum_i \mathrm{L}(\Phi_i)$ *such that*

$$\mathrm{R}(\Phi) = \sum_i \mathrm{R}(\Phi_i).$$

Proof: Deep Linear Combinations of Networks

Proof:

- Let Φ^{sum} and Φ^{diag} be the single-layer networks realizing the maps

$$\mathrm{sum}\colon \mathbb{R}^d \times \mathbb{R}^n \times \mathbb{R}^n \ni (x, y, z) \mapsto (x, y+z) \in \mathbb{R}^d \times \mathbb{R}^n,$$
$$\mathrm{diag}\colon \mathbb{R}^d \times \mathbb{R}^n \ni (x, y) \mapsto (x, x, y) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^n.$$

- Then the sum with skip connections

$$\mathbb{R}^d \times \mathbb{R}^n \ni (x, y) \mapsto (x, \mathrm{R}(\Phi_i)(x) + y) \in \mathbb{R}^d \times \mathbb{R}^n$$

  is realized by the network

$$\Psi_i := \Phi^{\mathrm{sum}} \bullet \mathrm{FP}\left(\Phi^{\mathrm{Id}}_{d,\mathrm{L}(\Phi_i)}, \Phi_i, \Phi^{\mathrm{Id}}_{N,\mathrm{L}(\Phi_i)}\right) \bullet \Phi^{\mathrm{diag}},$$

  which satisfies $\mathrm{B}(\Psi_i) \leq \max\{\mathrm{B}(\Phi_i), 1\}$, $\mathrm{W}(\Psi_i) \leq \mathrm{W}(\Phi_i) + 2d + 10$, $\mathrm{L}(\Psi_i) = \mathrm{L}(\Phi_i)$.

Proof: Deep Linear Combinations of Networks

- Let Φ^{pr} and Φ^{ins} be the single-layer networks realizing the maps

$$\begin{aligned} \mathrm{pr}\colon \mathbb{R}^d \times \mathbb{R}^n \ni (x,y) \mapsto y \in \mathbb{R}^n, \\ \mathrm{ins}\colon \mathbb{R}^d \ni x \mapsto (x,0) \in \mathbb{R}^d \times \mathbb{R}^n. \end{aligned}$$

- Then the network $\Phi := \Phi^{\mathrm{pr}} \bullet \Psi_1 \odot \cdots \odot \Psi_k \bullet \Phi^{\mathrm{ins}}$ has the desired properties. □

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How can the identity be realized using ReLU networks?
- Repetition: What is sparse concatenation, and how does it differ from non-sparse concatenation?
- Repetition: What are skip connections, what are they good for, and how can they be implemented using ReLU networks?
- Discussion: To what extent are the results of this video limited to ReLU networks?

Mathematics of Deep Learning, Summer Term 2020

Week 8, Video 2

ReLU Representation of Saw-Tooth Functions

Philipp Harms Lars Niemann

University of Freiburg

ReLU Representation of the Hat Function

Lemma

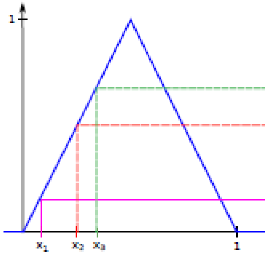
The *hat function*

$$F(x) := \rho_R(2x) - 2\rho_R(2x-1) + \rho_R(2x-2)$$

equals the *ReLU realization* *of the network* $\Phi^{\mathrm{hat}} := ((A_1, b_1), (A_2, 0))$ *with*

$$A_1 := (2,2,2)^\top, \quad b_1 := (0,-1,-2)^\top, \quad A_2 := (1,-2,1)\,.$$

This network satisfies $\mathrm{B}(\Phi^{\mathrm{hat}}) = 2$, $\mathrm{W}(\Phi^{\mathrm{hat}}) = 3$, *and* $\mathrm{L}(\Phi^{\mathrm{hat}}) = 2$.



ReLU Representation of Saw-Tooth Functions

Theorem

For any $n \in \mathbb{N}$, *the* *saw-tooth* *function* F_n *given by* $F_n(x) = 0$ *for* $x \notin (0, 1)$ *and*

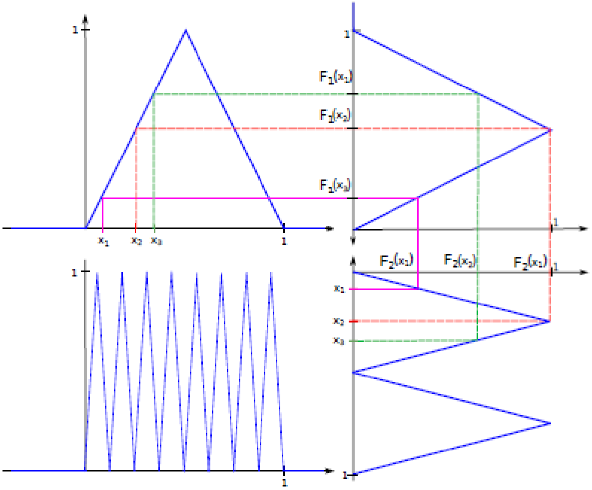
$$F_n(x) := \begin{cases} 2^n(x - i2^{-n}), & x \in [i2^{-n}, (i+1)2^{-n}],\ i \text{ even}, \\ 2^n((i+1)2^{-n} - x), & x \in [i2^{-n}, (i+1)2^{-n}],\ i \text{ odd}, \end{cases}$$

equals the *ReLU realization* *of the concatenated network* $\Phi_n := \bullet^n \Phi^{\mathrm{hat}}$ *with* $\mathrm{B}(\Phi_n) \leq 4$, $\mathrm{W}(\Phi_n) \leq 3$, *and* $\mathrm{L}(\Phi_n) = n + 1$.

Proof:

- F_n is the n-fold composition of hat functions.
- Thus, the n-fold concatenation $\bullet^n \Phi^{\mathrm{hat}}$ has the desired properties. □

Visualization of Saw-Tooth Functions



Top Left: F_1, Bottom Right: F_2, Bottom Left: F_4.

[Figure from Petersen, Ch. 3]

The Role of Depth

Remark: The theorem is surprising for the following reason:

- The realization of a shallow network Φ with two layers and input dimension 1 is piece-wise linear with at most $\mathrm{W}(\Phi)$ pieces.
- Similarly, networks of depth bounded by L have at most $\mathrm{W}(\Phi)^{L-1}$ pieces.
- In contrast, the previously introduced deep networks realize the saw-tooth function F_n, which has exponentially many pieces in $\mathrm{L}(\Phi)$.
- Thus, saw-tooth functions F_n can be represented very efficiently by deep networks, but not very efficiently by shallow networks.

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How can saw-tooth functions be represented by deep ReLU networks?
- Check: Why can the realization of a two-layer network Φ have at most $\mathrm{M}(\Phi)$ pieces?
- Check: Verify that the saw-tooth function is a composition of hat functions.
- Background: Can you show that the ReLU function is discriminatory?

Mathematics of Deep Learning, Summer Term 2020

Week 8, Video 3

Saw-Tooth Approximation of the Square Function

Philipp Harms   Lars Niemann

University of Freiburg

Saw-Tooth Approximation of the Square Function

Setting: Let F_n, $n \in \mathbb{N}$, denote the saw-tooth functions of Video 2.

Lemma

The piece-wise linear functions

$$H_n(x) := x - \sum_{k=1}^{n} F_k(x) 2^{-2k}, \qquad n \in \mathbb{N},\ x \in \mathbb{R},$$

approximate the square function at an exponential rate:

$$\sup_{x \in [0,1]} \left| x^2 - H_n(x) \right| \leq 2^{-2(n+1)}, \qquad n \in \mathbb{N}\,.$$

Remark: This makes us optimistic that, using sufficiently deep networks, we can approximate the square function efficiently.

Visualizing the Approximation of the Square Function

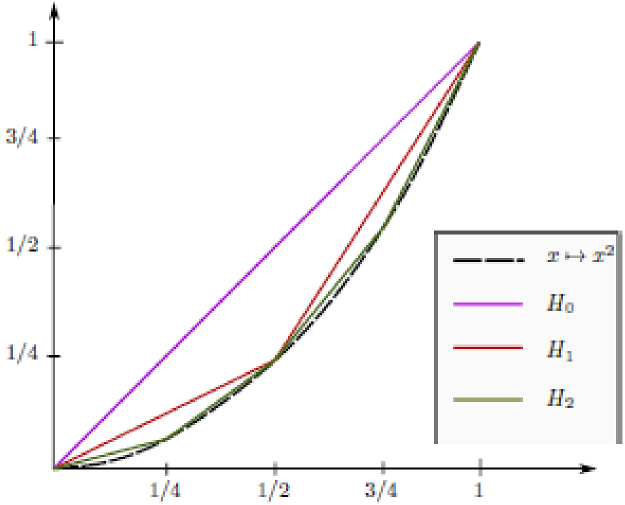


Figure: Approximants $H_n(x) := x - \sum_{k=1}^{n} F_k(x) 2^{-2k}$ of the square function x^2.
[Figure from Petersen, Ch. 3]

Proof: Approximating the Square Function by Saw-Tooths

Proof:

- By induction, the function H_n is piecewise linear with breakpoints $k2^{-n}$ for $k \in \{0, \dots, 2^n\}$, and $H_n(x) = x^2$ at the breakpoints.
- By convexity, $H_n(x) \geq x^2$ for $x \in [0, 1]$.
- For any x between the breakpoints $\ell := k2^{-n}$ and $u := (k+1)2^{-n}$,

$$\left|H_n(x) - x^2\right| = H_n(x) - x^2 = \frac{u - x}{u - \ell}\ell^2 + \frac{x - \ell}{u - \ell}u^2 - x^2.$$

- This quadratic function assumes its maximum at its unique critical point x^*, and one easily verifies that

$$x^* = \frac{u + \ell}{2}, \qquad H_n(x^*) - (x^*)^2 = \left(\frac{u - \ell}{2}\right)^2 = 2^{-2(n+1)}.$$

□

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How can the square function be approximated by linear combinations of saw-tooth functions?

- Check: Verify that a secant approximation of the square function is worst half-way between the abscissas of the intersection.

  Discussion: How could the saw-tooth approximation of the square function be implemented by ReLU networks. Spoiler alert: think about this before you watch the next video.

Mathematics of Deep Learning, Summer Term 2020

Week 8, Video 4

ReLU Approximation of Multiplication

Philipp Harms Lars Niemann

University of Freiburg

Approximating the Square Function

Remark: As an auxiliary result, we will approximate the square function by ReLU networks, building on the saw-tooth approximations of the square function.

Lemma

The square function can be approximated by ReLU networks at an exponential rate:

$$\forall n \in \mathbb{N}\ \exists \Phi : \mathrm{B}(\Phi) \leq 4, \mathrm{W}(\Phi) \leq 5, \mathrm{L}(\Phi) = n + 2,$$

$$\sup_{x \in [-1,1]} \left| x^2 - \mathrm{R}(\Phi)(x) \right| \leq 2^{-2(n+1)} .$$

Attempted Proof: Approximating the Square Function

Attempted proof: Strategy of Yarotsky (2017).

- Approximate the square function by saw-tooth functions: For any $n \in \mathbb{N}$,

$$\sup_{x\in[0,1]} \left|x^2 - H_n(x)\right| \leq 2^{-2(n+1)}, \qquad H_n(x) = x - \sum_{k\leq n} F_k 2^{-2k}.$$

- Represent each saw-tooth function by a network: $F_k = \mathrm{R}(\bullet^k \Phi^\wedge)$.
- Use skip connections to get networks of equal depth: $F_k = \mathrm{R}(\Phi_k)$ with $\Phi_k \coloneqq \Phi^{\mathrm{Id}}_{1,n-k} \odot \bullet^k \Phi^\wedge$.
- Take linear combinations of $\Phi_1, \ldots, \Phi_n$ to obtain networks of width proportional to n.
- Alternatively, using deep linear combinations, one obtains networks of depth proportional to n^2.
- In any case, this strategy is sub-optimal. □

Proof: Approximating the Square Function

Proof: Strategy of Perekrestenko e.a. (2018).

- As before, approximate the square by saw-tooth functions H_n:

$$\sup_{x\in[0,1]} \left|x^2 - H_n(x)\right| \le 2^{-2(n+1)}, \qquad H_n(x) = x - \sum_{k\le n} F_k 2^{-2k}.$$

- Recall that F_n is the n-fold composition of the hat function

$$F(x) := 2\rho_R(x) - 4\rho_R(x - \tfrac{1}{2}) + 2\rho_R(x-1),$$

and note that $H_n(x) = H_{n-1}(x) - 2^{-2n}F_n(x)$.

- This yields the recursion

$$\begin{cases} F_n(x) = 2\rho_R(F_{n-1}(x)) - 4\rho_R(F_{n-1}(x) - \tfrac{1}{2}) + 2\rho_R(F_{n-1}(x) - 1), \\ H_n(x) = \rho_R(H_{n-1}(x)) - \rho_R(-H_{n-1}(x)) - 2^{-2n}F_n(x), \end{cases}$$

where the term $F_n(x)$ on the right-hand side can be substituted by a term involving the functions $F_{n-1}(x)$ using the first equation.

Proof: Approximating the Square Function (cont.)

- Each recursive step corresponds to a network layer:

$$\begin{pmatrix} F_n \\ H_n \end{pmatrix} = W_1 \rho_R \left(W_2 \begin{pmatrix} F_{n-1} \\ H_{n-1} \end{pmatrix} \right),$$

$$W_1(x) = \begin{pmatrix} 2 & -2^{-2n+1} \\ -4 & 2^{-2n+2} \\ 2 & 2^{-2n+1} \\ 0 & 1 \\ 0 & -1 \end{pmatrix}^\top \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix},$$

$$W_2(x) = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 0 \\ 1/2 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

- Thus, using non-sparse concatenation $\bullet$, the iteration for H_n with $F_0(x) = |x|$ and $H_0(x) = |x|$ can be realized by a ReLU network Φ of depth $n+2$, width 5, and weights bounded by 4. □

Approximating Multiplication

Remark: The previous lemma on approximation of the square function implies the following theorem:

Theorem

Multiplication can be approximated by ReLU networks at an exponential rate:

$$\forall n \in \mathbb{N}\ \exists \Phi : \mathrm{B}(\Phi) \leq 8, \mathrm{W}(\Phi) \leq 10, \mathrm{L}(\Phi) = n + 2,$$

$$\sup_{x,y \in [-1,1]} |xy - \mathrm{R}(\Phi)(x,y)| \leq 2^{-2n-1}.$$

Remark: On domains $x, y \in [-K, K]$, the weight bound changes to a quadratic polynomial in K.

Proof: Approximating Multiplication

Proof:

- By polarization, we have for $x, y \in [-1,1]$ that

$$xy = \left(\frac{x+y}{2}\right)^2 - \left(\frac{x-y}{2}\right)^2. \qquad (*)$$

- Approximate the square function on $[-1,1]$ with precision $2^{-2(n+1)}$ by a neural network Φ_0 with $\mathrm{B}(\Phi_0) \le 4$, $\mathrm{W}(\Phi_0) \le 5$, and $\mathrm{L}(\Phi_0) = n+2$.
- Define neural networks Φ_1 and Φ_2 as

$$\Phi_1 := \left(\left(\frac{1}{2}\begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}, 0\right)\right), \qquad \Phi_2 := \Big(\big((1,-1),0\big)\Big).$$

- As the realization of $\Phi := \Phi_2 \bullet \mathrm{FP}(\Phi_0, \Phi_0) \bullet \Phi_1$ equals $(*)$ with squares replaced by $\mathrm{R}(\Phi_0)$, the error is at most 2^{-2n-1}. □

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How can multiplication be approximated by ReLU networks at an exponential rate?
- Transfer: Compare the ReLU approximation to the sigmoidal approximation of multiplication. See Week 3.
- Discussion: Using harmonic analysis we previously established polynomial upper bounds on network approximation rates—are they in contradiction to the exponential approximation rate established here?

Mathematics of Deep Learning, Summer Term 2020

Week 8, Video 5

ReLU Approximation of Analytic Functions

Philipp Harms   Lars Niemann

University of Freiburg

Approximating Monomials

Lemma

Monomials can be approximated by ReLU networks at an exponential rate:

$\forall d, p, n \in \mathbb{N}\ \forall i_1, \ldots, i_p \in \{1, \ldots, d\}\ \exists \Phi :$
$\mathrm{B}(\Phi) \leq 8, \mathrm{W}(\Phi) \leq 2d + 10, \mathrm{L}(\Phi) = p(n+2),$

$$\sup_{x \in [-1,1]^d} \left| x_{i_1} \cdots x_{i_p} - \mathrm{R}(\Phi) \right| (x) \leq 2^{-2n-1}$$

Remark:

- Via dictionary learning, this leads to optimal polynomial approximation rates for many signal classes.
- More interestingly, in contrast to our previous results, it also leads to exponential approximation rates for real-analytic functions, including e.g. sinusoidal functions and oscillatory textures.

Proof: Approximating Monomials

Proof:

- For any $i \in \{1, \dots, d\}$, the multiplication with skip connections

$$(x_1, \dots, x_d, y) \mapsto (x_1, \dots, x_d, x_i y)$$

can be approximated by a network Ψ_i with $\mathrm{B}(\Psi_i) \le 8$, $\mathrm{W}(\Psi_i) \le 2d + 10$, $\mathrm{L}(\Psi_i) = n + 2$, and

$$\sup_{x_1,\dots,x_d,y \in [-1,1]} \|(x_1, \dots, x_d, x_i y) - \mathrm{R}(\Psi_i)(x_1, \dots, x_d, y)\|_\infty \le 2^{-2n-1}.$$

- As the realizations of Ψ_i are 1-Lipschitz and bounded by 1, the net

$$\Phi := (((0_{(\mathbb{R}^d)^*}, 1), 0)) \bullet \Psi_{i_1} \odot \cdots \odot \Psi_{i_p} \bullet \left(\left(\begin{pmatrix} \mathrm{Id}_{\mathbb{R}^d} \\ 0_{(\mathbb{R}^d)^*} \end{pmatrix}, \begin{pmatrix} 0_{\mathbb{R}^d} \\ 1 \end{pmatrix} \right) \right)$$

satisfies $\mathrm{B}(\Phi) \le 8$, $\mathrm{W}(\Phi) \le 2d + 10$, $\mathrm{L}(\Phi) = p(n+3)$, and

$$\sup_{x_1,\dots,x_d \in [-1,1]} \left| x_{i_1} \cdots x_{i_p} - \mathrm{R}(\Phi)(x_1, \dots, x_d) \right| \le 2^{-2n-1}. \qquad \square$$

Real-Analytic Functions

Definition

A function $f\colon (-r,r)^d \to \mathbb{R}$ is real-analytic if it is given by a power series

$$f(x) = \sum_{k \in \mathbb{N}^d} a_k x^k, \qquad x \in (-r,r)^d,$$

for some coefficients $(a_k)_{k \in \mathbb{N}^d}$.

Remark:

- The power series converges absolutely on $(-r,r)^d$.
- Thus, if $r > 1$, then a is summable, i.e., $\|a\|_{\ell^1} \coloneqq \sum_{k \in \mathbb{N}^d} |a_k| < \infty$.

Approximating Real-Analytic Functions

Theorem

Real-analytic functions can be approximated by ReLU networks:

$$\forall d \in \mathbb{N}_{\geq 2}\ \forall \delta > 0\ \exists \bar{\epsilon} > 0\ \forall \epsilon \in (0, \bar{\epsilon})\ \forall (a_k)_{k \in \mathbb{N}^d} \in \ell^1\ \exists \Phi :$$

$$\mathrm{B}(\Phi) \leq 8 \sum_{k \in \mathbb{N}^d} |a_k|, \mathrm{W}(\Phi) \leq (2d + 10), \mathrm{L}(\Phi) \leq \left(e\left(\tfrac{1}{d\delta} \log_2 \tfrac{1}{\epsilon} + 1 \right) \right)^{2d},$$

$$\sup_{x \in [-1+\delta, 1-\delta]^d} \left| \sum_{k \in \mathbb{N}^d} a_k x^k - \mathrm{R}(\Phi)(x) \right| \leq 2\epsilon \|a_k\|_{\ell^1}.$$

Remark: Note that the error decays exponentially in $L^{1/(2d)}$ because

$$L(\Phi) \leq \left(e\left(\tfrac{1}{d\delta} \log_2 \tfrac{1}{\epsilon} + 1 \right) \right)^{2d} \Leftrightarrow \epsilon \leq \exp(-d\delta(e^{-1} L^{1/(2d)} - 1)).$$

Approximating Real-Analytic Functions

Proof:

- Without loss of generality, $\|a_k\|_{\ell^1} = 1$.
- Truncation: Let $p := \lceil \frac{1}{\delta} \log_2 \frac{1}{\epsilon} \rceil$, $f(x) := \sum_{k \in \mathbb{N}^d} a_k x^k$, $f_p(x) := \sum_{k \in \mathbb{N}^d_{\leq p}} a_k x^k$. Then

$$\sup_{x \in [-1+\delta, 1-\delta]^d} |f(x) - f_p(x)| \leq (1-\delta)^p \leq \epsilon.$$

- Monomial approximation: Let $n := \lceil \frac{1}{2} \log_2 \frac{1}{\epsilon} \rceil$. Approximate each monomial x^k by a network Φ_k with $\mathrm{B}(\Phi) \leq 8$, $\mathrm{W}(\Phi) \leq 2d + 10$, $\mathrm{L}(\Phi_k) = p(n+2)$, and

$$\sup_{x \in [-1,1]^d} \left| x^k - \mathrm{R}(\Phi_k)(x) \right| \leq 2^{-2n-1} \leq \epsilon.$$

Approximating Real-Analytic Functions

- Deep linear combinations of the $\binom{p+d}{d}$ monomials: there is a network Φ with $\mathrm{B}(\Phi) \leq 8$, $\mathrm{W}(\Phi) \leq 2d + 11$, $\mathrm{L}(\Phi) = p(n+2)\binom{p+d}{d}$,

$$\sup_{x \in [-1,1]^d} |f_p(x) - \mathrm{R}(\Phi)(x)| \leq \epsilon.$$

- Depth bound: for sufficiently small $\bar{\epsilon}$ and $\epsilon < \bar{\epsilon}$,

$$\begin{aligned}\mathrm{L}(\Phi) &= p(n+2)\binom{p+d}{d} = p(n+2)\frac{(p+d)\cdots(p+1)}{d!} \\ &\leq p(n+2)\left(\frac{p+d}{d/e}\right)^d = p(n+2)\left(e(\tfrac{p}{d}+1)\right)^d \\ &\leq \left(e(\tfrac{1}{d\delta}\log_2\tfrac{1}{\epsilon}+1)\right)^{2d},\end{aligned}$$

where the last inequality follows by an elementary calculation from the definitions of p and n and the assumption $d \geq 2$. □

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How can real-analytic functions be approximated by ReLU networks at an exponential rate?
- Background: What is the difference between smooth, real-analytic, and holomorphic functions?
- Check: Prove the inequality $d! \geq (d/e)^d$, which was used in the last proof. Hint: $d^d/d!$ is a summand in the series expansion of e^d.
- Discussion: Can real-analytic functions be approximated by shallow networks at an exponential rate?
- Transfer: What other assumptions on the signal class besides real analyticity might increase the approximation rate?

Mathematics of Deep Learning, Summer Term 2020

Week 8, Video 6

Wrapup

Philipp Harms   Lars Niemann

University of Freiburg

Outlook on this week's discussion and reading session

- Reading:
  - Yarotsky (2017): Error bounds for approximations with deep ReLU networks. Neural Networks 94, pp. 103–114.
  - Perekrestenko, Grohs, Elbrächter, Bölcskei (2018): The universal approximation power of finite-width deep ReLU Networks. `arXiv:1806.01528`
  - E, Wang (2018): Exponential convergence of the deep neural approximation for analytic functions. `arXiv:1807.00297`

Summary by learning goals

Having heard this lecture, you can now . . .

- Establish exponential rates for the approximation of real-analytic functions by deep ReLU networks.
- Explain the role of skip connections in this construction.

Review and Outlook

- Topics covered in this lecture series:
  - Statistical learning theory
  - Universal approximation theorems
  - Dictionary learning
  - Kolmogorov–Arnold representation
  - Harmonic analysis
  - Information theory
  - ReLU networks and the role of depth
- Topics not covered in this lecture series: (non-exhaustive)
  - Residual, recurrent, and adversarial networks; auto-encoders
  - Manifold assumptions on the data distribution
  - Generalization capability and implicit regularization
  - Many practical issues