Fundamental Belief: Universal Approximation Theorems

Ju Sun
Computer Science & Engineering
University of Minnesota, Twin Cities

September 13, 2022

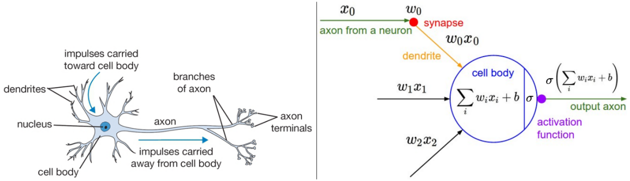
Logistics

- HW 0 posted (due: midnight Sep 24)
    * Problem 1 (Chain rules, gradient and Hessian; $4/15$)
    * Problem 2 (Matrix norms, inner products, traces; $7/15$)
    * Problem 3 (Taylor expansion of multivariate functions; $2.5/15$)
    * Problem 4 (Conditional probability and Bayes' Rule; $1.5/15$)
- Change in office hours (all office hours in **hybrid mode**)
    * Ju: 2–4pm Mon
    * Hengkang: 1–3pm Thur
    * Yash: 12-2pm Fri

  We welcome questions on Piazza, which we'll check and answer frequently
- MSI (Minnesota Supercomputing Institute) class account expected later this week

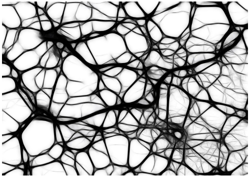
Outline

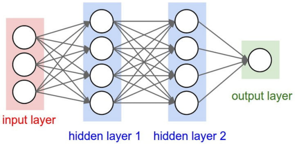
Recap I



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

biological neuron vs. **artificial neuron**



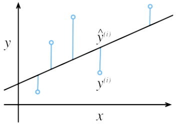


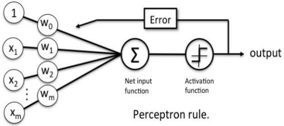
biological NN vs. **artificial NN**

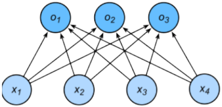
Artificial NN: (over)-simplification on **neuron** & **connection** levels

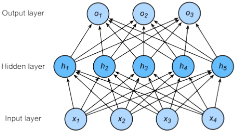
Recap II

Zoo of NN models in ML







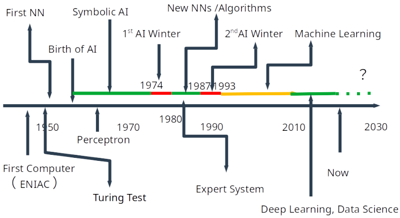


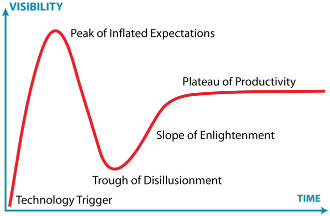
- Linear regression
- Perception and Logistic regression
- Softmax regression
- Multilayer perceptron (feedforward NNs)

Also:

- Support vector machines (SVM)
- PCA (autoencoder)
- Matrix factorization

Recap III





Brief history of NNs:

- 1943: first NNs invented (McCulloch and Pitts)
- 1958 –1969: perceptron (Rosenblatt)
- 1969: *Perceptrons* (Minsky and Papert)—end of perceptron
- 1980's–1990's: Neocognitron, CNN, back-prop, SGD—we use today
- 1990's–2010's: SVMs, Adaboosting, decision trees and random forests
- 2010's–now: DNNs and deep learning

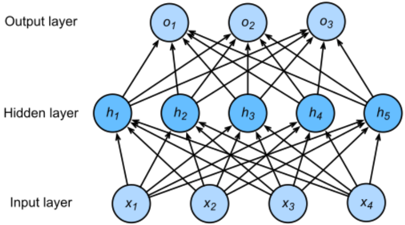
Outline

Supervised learning

Step	General view	NN view
1	Gather training set $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$	Gather training set $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$
2	Choose a family of functions, e.g., $\mathcal{H}$, so that there is an $f \in \mathcal{H}$ to ensure $\boldsymbol{y}_i \approx f(\boldsymbol{x}_i)$, $\forall\, i$	Choose a NN with k neurons, so that there is a group of weights $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)$ ensuring $\boldsymbol{y}_i \approx \{\mathrm{NN}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)\}(\boldsymbol{x}_i)$, $\forall i$
3	Set up a loss function ℓ	Set up a loss function ℓ
4	Find an $f \in \mathcal{H}$ to minimize the average loss $$\frac{1}{n}\sum_{i=1}^{n} \ell\left(\boldsymbol{y}_i, f(\boldsymbol{x}_i)\right)$$	Find weights $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)$ to minimize the average loss $$\frac{1}{n}\sum_{i=1}^{n} \ell\left[\boldsymbol{y}_i, \{\mathrm{NN}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)\}(\boldsymbol{x}_i)\right]$$

Why we trust NNs? They're "powerful"—encoding "large" $\mathcal{H}$

Three fundamental questions in DL



- k-layer NNs: with k layers of weights (along the deepest path)
- k-hidden-layer NNs: with k hidden layers of nodes (i.e., $(k+1)$-layer NNs)

- **Approximation**: is it powerful, i.e., the $\mathcal{H}$ large enough for all possible weights? (now)
- **Optimization**: how to solve

$$\min_{\boldsymbol{w}_i' s, \boldsymbol{b}_i' s} \frac{1}{n} \sum_{i=1}^{n} \ell \left[\boldsymbol{y}_i, \{\mathsf{NN}\left(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k\right)\} \left(\boldsymbol{x}_i\right)\right]$$

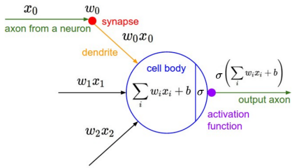
(later this course)

- **Generalization**: does the learned NN work well on "similar" data? (CSCI5525, and Deep Learning Theory)

Is NN powerful? first trial

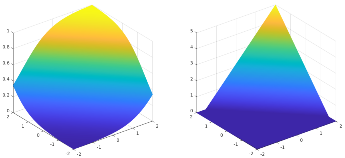
Think of single-output (i.e., $\mathbb{R}^n \mapsto \mathbb{R}$) problems first

A single neuron



$\mathcal{H} : \{\boldsymbol{x} \mapsto \sigma\left(\boldsymbol{w}^\intercal \boldsymbol{x} + b\right)\}$

- σ identity or linear: linear functions
- σ sign function $\operatorname{sign}\left(\boldsymbol{w}^\intercal \boldsymbol{x} + b\right)$ (perceptron): $0/1$ function with hyperplane threshold
- $\sigma = \frac{1}{1+e^{-z}}$: $\left\{\boldsymbol{x} \mapsto \frac{1}{1+e^{-(\boldsymbol{w}^\intercal \boldsymbol{x}+b)}}\right\}$
- $\sigma = \max(0, z)$ (ReLU): $\{\boldsymbol{x} \mapsto \max(0, \boldsymbol{w}^\intercal \boldsymbol{x} + b)\}$

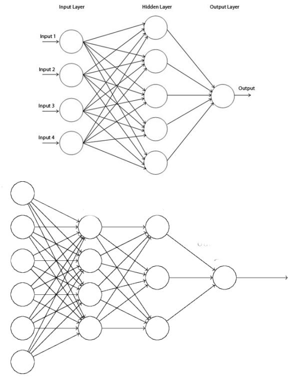


Question: What cannot be done?

Is NN powerful? second trial

Think of single-output (i.e., $\mathbb{R}^n \mapsto \mathbb{R}$) problems first

Add depth!



. . .

But make all hidden-nodes activations identity or linear

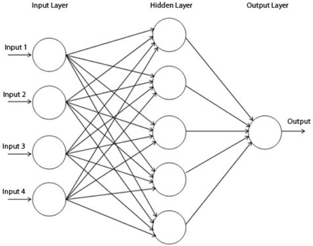
$$\sigma\left(\boldsymbol{w}_L^\intercal\left(\boldsymbol{W}_{L-1}\left(\ldots\left(\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1\right)+\ldots\right)\boldsymbol{b}_{L-1}\right)+b_L\right)$$

No better than a single neuron! Why?

Is NN powerful? third trial

Think of single-output (i.e., $\mathbb{R}^n \mapsto \mathbb{R}$) problems first

Add both depth & nonlinearity!



two-layer network, linear activation at output

Surprising news: **universal approximation theorem (UAT)**

The 2-layer network can approximate **arbitrary** continuous functions **arbitrarily** well, provided that the hidden layer is **sufficiently wide**.

— so we don't worry about limitation in the capacity

Outline

Why could UAT hold?

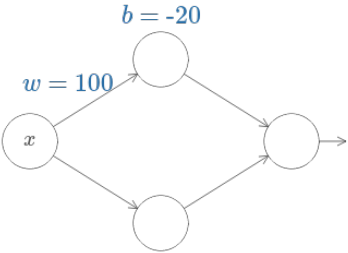
Visual "proof"

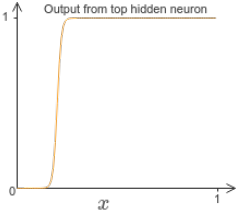
(http://neuralnetworksanddeeplearning.com/chap4.html)

Think of $\mathbb{R} \to \mathbb{R}$ functions first, $\sigma = \frac{1}{1+e^{-z}}$

- Step 1: Build "step" functions
- Step 2: Build "bump" functions
- Step 3: Sum up bumps to approximate

Step 1: build step functions

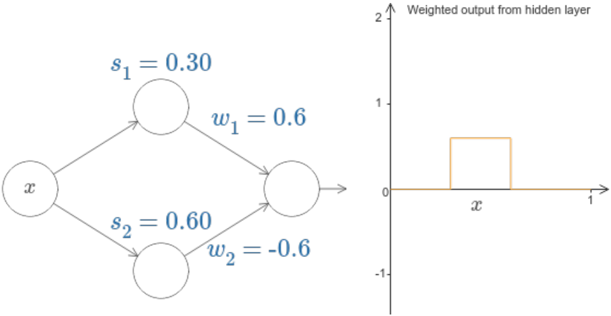




$$y = \frac{1}{1 + e^{-(wx+b)}} = \frac{1}{1 + e^{-w(x-b/w)}}$$

- Larger w, sharper transition
- Transition around $-b/w$, written as s

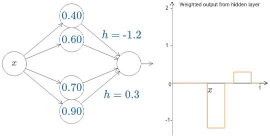
Step 2: build bump functions



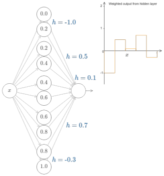
$$0.6 * \mathrm{step}(0.3) - 0.6 * \mathrm{step}\,(0.6)$$

Write h as the bump height

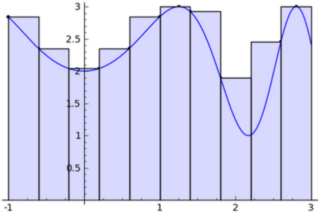
two bumps



five bumps



ultimate idea ... familiar?



Message: all $\mathbb{R} \mapsto \mathbb{R}$ functions can be "well" approximated using 2-layer NN's

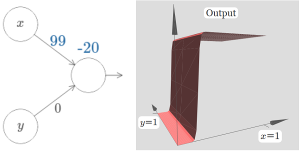
What about high-dimensional?

Similar story

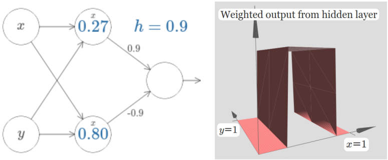
- Step 1: Build "step" functions
- Step 2: Build "bump" functions
- Step 3: Build "tower" functions
- Step 4: Sum up bumps to approximate

http://neuralnetworksanddeeplearning.com/chap4.html

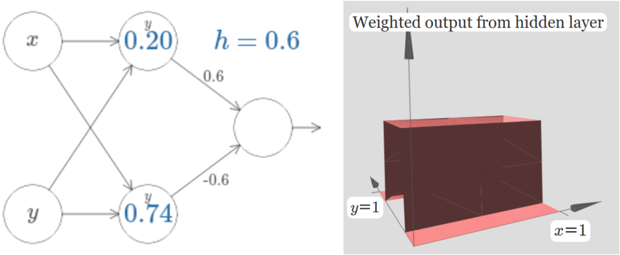
Steps 1 & 2: build step and bump functions



step in x by setting large weight for x

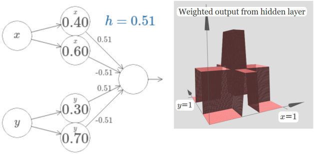


bump in x by diff of two steps in x

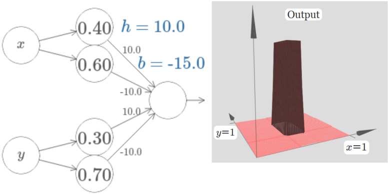


bump in y by diff of two steps in y

Step 3: build tower functions

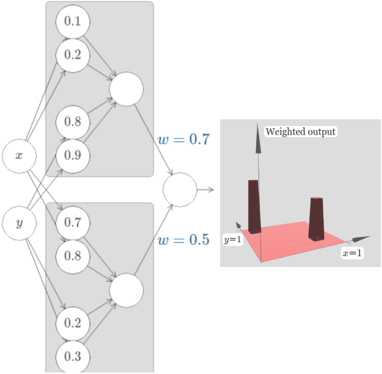


sum up x, y bumps to obtain a stair tower

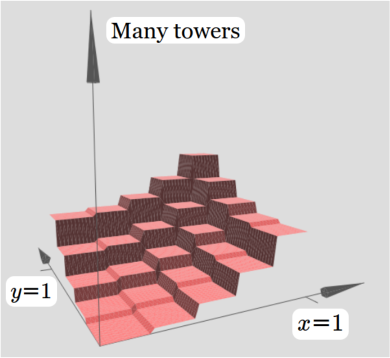


threshold to obtain a sharp tower

Step 4: sum up towers for approximation



sum up two towers



sum up many towers

Message: all $\mathbb{R}^2 \mapsto \mathbb{R}$ functions can be "well" approximated using 3-**layer** NN's Question: Possible using 2-layer NNs only?

General cases?

- What about $\mathbb{R}^n \mapsto \mathbb{R}$ functions?
  The "step $\rightarrow$ (bump) $\rightarrow$ tower $\rightarrow$ tower array" construction carries over
- What about $\mathbb{R}^n \mapsto \mathbb{R}^m$ functions?
  Approximate each $\mathbb{R}^n \mapsto \mathbb{R}$ separately and then glue them together

Message: All $\mathbb{R}^n \mapsto \mathbb{R}^m$ functions can be "well" approximated using 2-layer NN's

Outline

[A] universal approximation theorem (UAT)

Theorem (UAT, [Cybenko, 1989, Hornik, 1991])

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that we may define:

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} v_i \sigma\left(\boldsymbol{w}_i^T \boldsymbol{x} + b_i\right) = \boldsymbol{v}^\intercal \sigma\left(\boldsymbol{W}^\intercal \boldsymbol{x} + \boldsymbol{b}\right)$$

as an approximate realization of the function f; that is,

$$|F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon$$

for all $\boldsymbol{x} \in I_m$.

Rigorous proof?

The proof is very technical ... functional analysis

1. Riesz Representation: Every linear functional on $C^0([0,1]^k)$ is given by
$$f \mapsto \int_{[0,1]^k} f(x) d\mu(x), \qquad \mu \in \mathcal{M}$$
where $\mathcal{M} = \left\{\text{finite signed regular Borel measures on } [0,1]^k\right\}$.

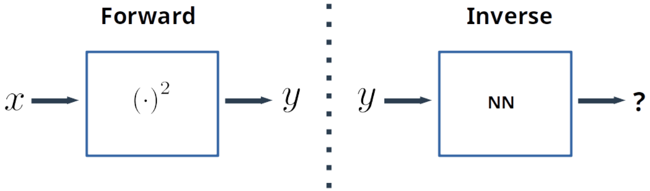
2. **Lemma.** Suppose for each $\mu \in \mathcal{M}$, we have
$$\int_{[0,1]^k} \phi(w \cdot x + b) d\mu(x) = 0 \quad \forall w, b \quad \Rightarrow \quad \mu = 0. \quad (0.1)$$
Then $\mathrm{Nets}_1(\phi)$ is dense in $C^0([0,1]^k)$.

3. **Lemma.** ϕ continuous, sigmoidal $\quad \Rightarrow \quad$ satisfies (0.1).

Thoughts on UAT

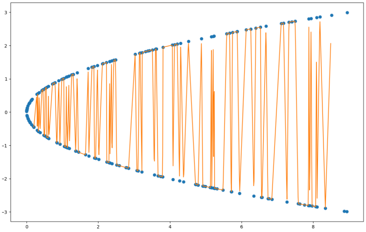
- $\sigma : \mathbb{R} \to \mathbb{R}$ **be a nonconstant, bounded, and continuous**: what about ReLU (leaky ReLU) or sign function (as in perceptron)? We have many UAT **theorem(s)**
- I_m **denote the m-dimensional unit hypercube** $[0, 1]^m$: this can replaced by any compact subset of $\mathbb{R}^m$
- **there exist an integer** N: but how large N needs to be? (later)
- **The space of real-valued continuous functions on** I_m: two examples to ponder on
  - binary classification
  - learn to solve square root

Learn to take square-root



Suppose we lived in a time square-root is not defined ...

- Training data: $\{x_i, x_i^2\}_i$, where $x_i \in \mathbb{R}$
- Forward: if $x \mapsto y$, $-x \mapsto y$ also
- To invert, what to output? What if just throw in the training data?



Thoughts

- Approximate continuous functions with vector outputs, i.e., $I_m \to \mathbb{R}^n$? think of the component functions
- Map to $[0, 1]$, $\{-1, +1\}$, $[0, \infty)$? choose appropriate activation σ at the output

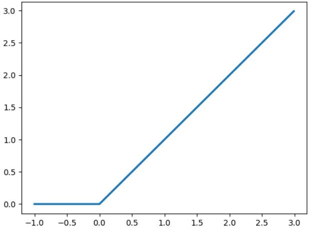
$$F(x) = \sigma\left(\sum_{i=1}^{N} v_i \sigma\left(\boldsymbol{w}_i^T \boldsymbol{x} + b_i\right)\right)$$

  ... universality holds in modified form
- Get deeper? three-layer NN? change to matrix-vector notation for convenience

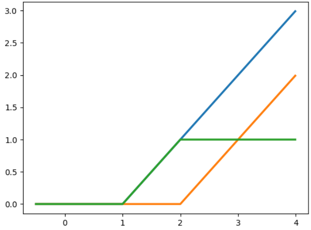
$$F(x) = \boldsymbol{w}^\intercal \sigma(\boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) \quad \text{as} \sum_k w_k g_k(\boldsymbol{x})$$

  use w_k's to linearly combine the same function
- **For geeks**: approximate both f and f'? check out [Hornik et al., 1990]

What about ReLU?



ReLU



difference of ReLU's

what happens when the slopes of the ReLU's are changed?

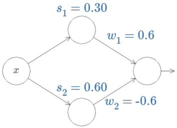
How general σ can be? ... enough when σ not a polynomial [Leshno et al., 1993, Gühring et al., 2020, DeVore et al., 2021]

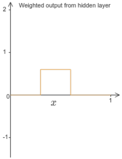
Outline

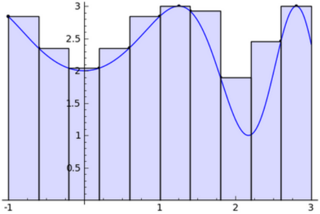
What's bad about shallow NNs?

From UAT, "... there exist an interger N, ...", but how large?

What happens in $1D$?







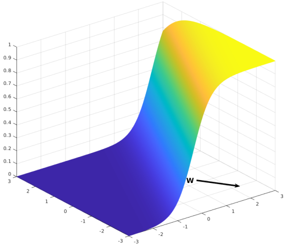
Assume the target f is 1-Lipschitz, i.e., $|f(x) - f(y)| \leq |x - y|, \forall\, x, y \in \mathbb{R}$

For ε accuracy, need $\frac{1}{\varepsilon}$ bumps

What's bad about shallow NNs?

From UAT, "... there exist an interger N, ...", but how large?

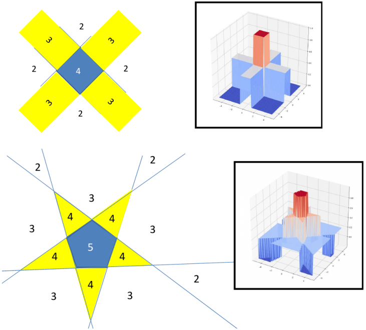
What happens in $2D$? Visual proof in 2D first



$\sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + \boldsymbol{b})$, σ sigmod

approach 2D step function when

making $\boldsymbol{w}$ large



Credit: CMU 11-785

Visual proof for 2D functions

Keep increasing the number of step functions that are distributed evenly ...

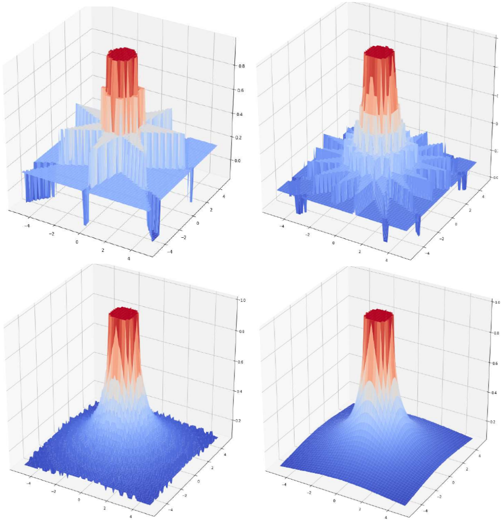


Image Credit: CMU 11-785

From UAT, "... there exist an interger N, ...", but how large?

What happens in $2D$?

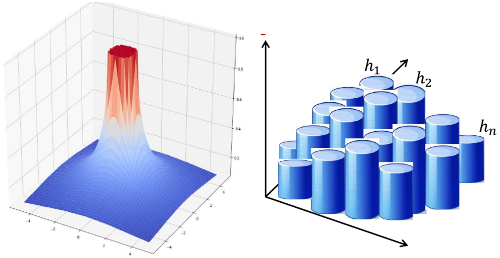


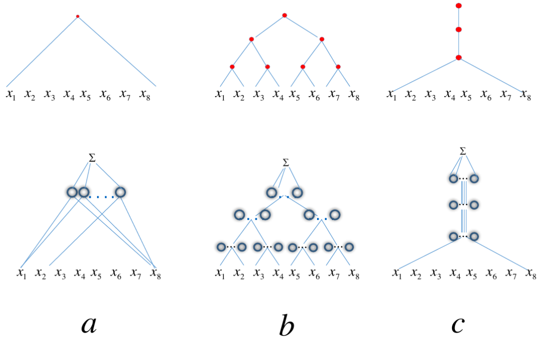
Image Credit: CMU 11-785

Assume the target f is 1-Lipschitz, i.e., $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le \|\boldsymbol{x} - \boldsymbol{y}\|_2, \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^2$

For ε accuracy, need $O\left(\varepsilon^{-2}\right)$ bumps. What about the n-D case? $O(\varepsilon^{-n})$.

What's good about deep NNs?

- Learn Boolean functions ($f : \{+1, -1\}^n \mapsto \{+1, -1\}$): DNNs can have #nodes linear in n, whereas 2-layer NN needs exponential nodes (more in HW1)
- What general functions set deep and shallow NNs apart?



A family: compositional function [Poggio et al., 2017]

Compositional functions

$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))) \qquad (4)$$

W_m^n: class of n-variable functions with partial derivatives up to m-th order,
$W_m^{n,2} \subset W_m^n$ is the compositional subclass following binary tree structures

Theorem 1. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not a polynomial. For $f \in W_m^n$ the complexity of shallow networks that provide accuracy at least ϵ is*

$$N = \mathcal{O}(\epsilon^{-n/m}) \text{ and is the best possible.} \qquad (5)$$

Theorem 2. *For $f \in W_m^{n,2}$ consider a deep network with the same compositonal architecture and with an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ which is infinitely differentiable, and not a polynomial. The complexity of the network to provide approximation with accuracy at least ϵ is*

$$N = \mathcal{O}((n-1)\epsilon^{-2/m}). \qquad (6)$$

from [Poggio et al., 2017] ; see Sec 4.2 of [Poggio et al., 2017] for lower bound

Nonsmooth activation

A terse version of UAT

Proposition 2. *Let $\sigma =: \mathbb{R} \to \mathbb{R}$ be in $\mathcal{C}^0$, and not a polynomial. Then shallow networks are dense in $\mathcal{C}^0$.*

Shallow vs. deep

Theorem 4. *Let f be a L-Lipshitz continuous function of n variables. Then, the complexity of a network which is a linear combination of ReLU providing an approximation with accuracy at least ϵ is*

$$N_s = \mathcal{O}\left(\left(\frac{\epsilon}{L}\right)^{-n}\right),$$

wheres that of a deep compositional architecture is

$$N_d = \mathcal{O}\left(\left(n-1)(\frac{\epsilon}{L}\right)^{-2}\right).$$

from [Poggio et al., 2017]

Width-bounded DNNs

Narrower than $n + 4$ is fine

Theorem 1 (Universal Approximation Theorem for Width-Bounded ReLU Networks). *For any Lebesgue-integrable function $f : \mathbb{R}^n \to \mathbb{R}$ and any $\epsilon > 0$, there exists a fully-connected ReLU network $\mathscr{A}$ with width $d_m \leq n + 4$, such that the function $F_{\mathscr{A}}$ represented by this network satisfies*

$$\int_{\mathbb{R}^n} |f(x) - F_{\mathscr{A}}(x)| \mathrm{d}x < \epsilon. \tag{3}$$

But no narrower than $n - 1$

Theorem 3. *For any continuous function $f : [-1, 1]^n \to \mathbb{R}$ which is not constant along any direction, there exists a universal $\epsilon^* > 0$ such that for any function F_A represented by a fully-connected ReLU network with width $d_m \leq n - 1$, the L^1 distance between f and F_A is at least ϵ^*:*

$$\int_{[-1,1]^n} |f(x) - F_A(x)| \mathrm{d}x \geq \epsilon^*. \tag{5}$$

from [Lu et al., 2017]; see also [Kidger and Lyons, 2019]

Deep vs. shallow still active area of research

Number one principle of DL

Fundamental theorem of DNNs

Universal approximation theorems (UATs)

Fundamental slogan of DL

Where there is a function, there is a NN...
and go ahead fitting it!

Outline

Suggested reading

- Chap 4, Neural Networks and Deep Learning (online book) http://neuralnetworksanddeeplearning.com/chap4.html
- Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. (by Poggio et al) https://arxiv.org/abs/1611.00740
- Expressivity of Deep Neural Networks (by Ingo Gühring, Mones Raslan, Gitta Kutyniok) https://arxiv.org/abs/2007.04759

[Cybenko, 1989] Cybenko, G. (1989). **Approximation by superpositions of a sigmoidal function.** *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

[DeVore et al., 2021] DeVore, R., Hanin, B., and Petrova, G. (2021). **Neural network approximation.** *Acta Numerica*, 30:327–444.

[Gühring et al., 2020] Gühring, I., Raslan, M., and Kutyniok, G. (2020). **Expressivity of deep neural networks.** *arXiv:2007.04759*.

[Hornik, 1991] Hornik, K. (1991). **Approximation capabilities of multilayer feedforward networks.** *Neural Networks*, 4(2):251–257.

[Hornik et al., 1990] Hornik, K., Stinchcombe, M., and White, H. (1990). **Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks.** *Neural Networks*, 3(5):551–560.

[Kidger and Lyons, 2019] Kidger, P. and Lyons, T. (2019). **Universal approximation with deep narrow networks.** *arXiv:1905.08539*.

[Leshno et al., 1993] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). **Multilayer feedforward networks with a nonpolynomial activation function can approximate any function.** *Neural Networks*, 6(6):861–867.

References ii

[Lu et al., 2017] Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). **The expressive power of neural networks: A view from the width.** In *Advances in neural information processing systems*, pages 6231–6239.

[Poggio et al., 2017] Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. (2017). **Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review.** *International Journal of Automation and Computing*, 14(5):503–519.