Universality / approximation error

+ generalization in deep learning

CPSC 532D: Modern Statistical Learning Theory

5 December 2022

cs.ubc.ca/~dsuth/532D/22w1/

Admin stuff:

- Office hours this week: up on Piazza later today, probably

  Tues 10am-noon online

  Weds 2pm-class } tbd online or hybrid

  Friday noon-1:30pm }

  + Milad's 4-5pm hybrid

- Details on final soon: "Kind of question," list of eligible topics

  1 sheet handwritten notes allowed

- If handing in both a4 and a5,

  hand in one by Sat and one by Wed 14th

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\quad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\quad f(x) = f^{(L)}(x)$

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
    - $f^{(0)}(x) = x$ $\quad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\quad f(x) = f^{(L)}(x)$
    - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\quad b_\ell \in \mathbb{R}^{d'_\ell}$ $\quad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\quad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\quad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\quad b_\ell \in \mathbb{R}^{d'_\ell}$ $\quad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\qquad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\qquad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\qquad b_\ell \in \mathbb{R}^{d'_\ell}$ $\qquad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**
  - Usually $\sigma_L(x) = x$; intermediate layers called **hidden layers**

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\qquad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\qquad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\qquad b_\ell \in \mathbb{R}^{d'_\ell}$ $\qquad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**
  - Usually $\sigma_L(x) = x$; intermediate layers called **hidden layers**
  - Common choices for **activations** σ:

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\quad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\quad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\quad b_\ell \in \mathbb{R}^{d'_\ell}$ $\quad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**
  - Usually $\sigma_L(x) = x$; intermediate layers called **hidden layers**
  - Common choices for **activations** σ:
    - Componentwise: $\mathrm{ReLU}(z) = \max\{z,0\}, \mathrm{sigmoid}(z) = 1/(1+\exp(-z))$

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\quad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\quad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\quad b_\ell \in \mathbb{R}^{d'_\ell}$ $\quad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**
  - Usually $\sigma_L(x) = x$; intermediate layers called **hidden layers**
  - Common choices for **activations** σ:
    - Componentwise: $\mathrm{ReLU}(z) = \max\{z, 0\}, \mathrm{sigmoid}(z) = 1/(1 + \exp(-z))$
    - $\mathrm{softmax}(z)_i = \exp(z_i)/\sum_j \exp(z_j)$, max pooling, attention, …

Deep learning

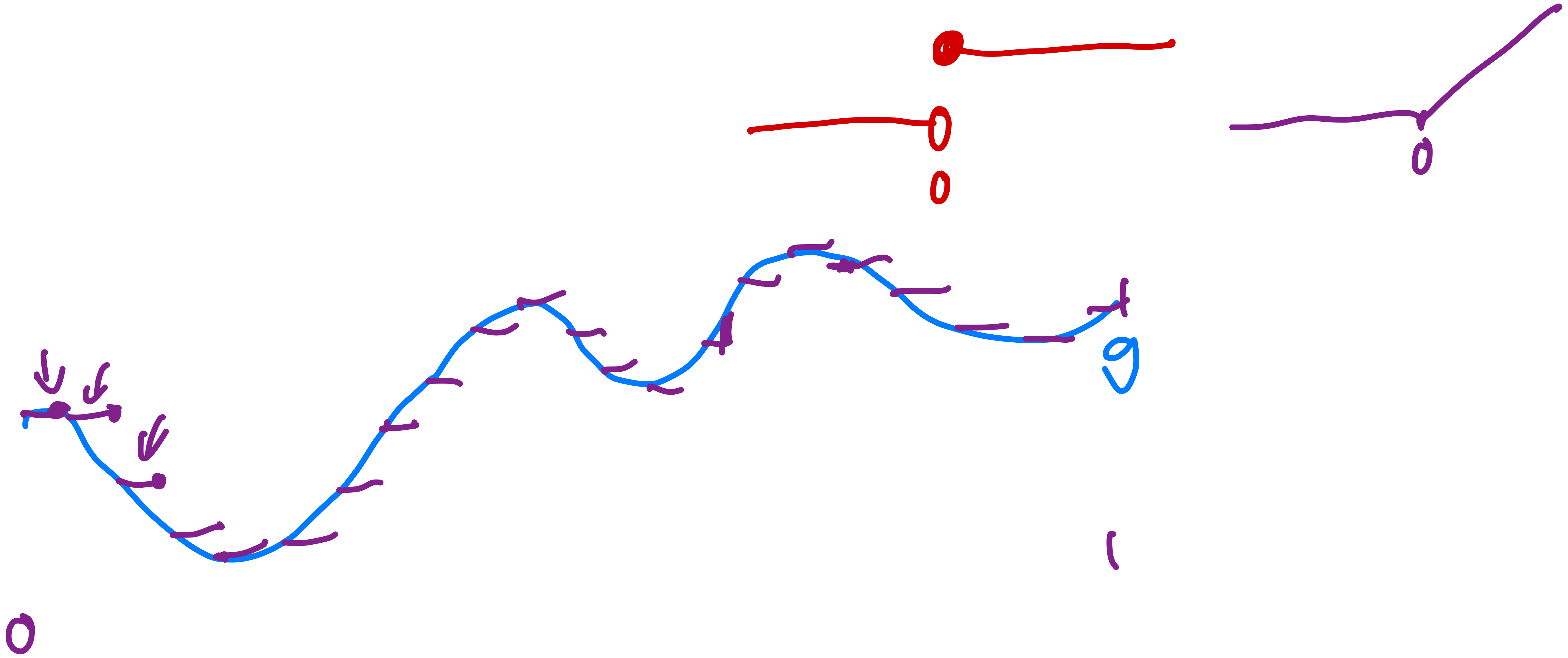
- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\qquad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\qquad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\qquad b_\ell \in \mathbb{R}^{d'_\ell}$ $\qquad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**
  - Usually $\sigma_L(x) = x$; intermediate layers called **hidden layers**
  - Common choices for **activations** σ:
    - Componentwise: $\mathrm{ReLU}(z) = \max\{z, 0\}, \mathrm{sigmoid}(z) = 1/(1 + \exp(-z))$
    - $\mathrm{softmax}(z)_i = \exp(z_i) / \sum_j \exp(z_j)$, max pooling, attention, …
  - Usually train via SGD, but it's **non-convex**: in general, possibility of local minima

Deep learning

- Mostly assuming **fully-connected**, **feedforward** nets ("multilayer perceptrons"):
  - $f^{(0)}(x) = x$ $\quad f^{(\ell)}(x) = \sigma_\ell(W_\ell f^{(\ell-1)}(x) + b_\ell)$ $\quad f(x) = f^{(L)}(x)$
  - $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ $\quad b_\ell \in \mathbb{R}^{d'_\ell}$ $\quad \sigma_\ell : \mathbb{R}^{d'_\ell} \to \mathbb{R}^{d_\ell}$ (usually $d'_\ell = d_\ell$)
  - Can think of this as a directed, acyclic computation graph, organized in **layers**
  - Usually $\sigma_L(x) = x$; intermediate layers called **hidden layers**
  - Common choices for **activations** σ:
    - Componentwise: $\mathrm{ReLU}(z) = \max\{z, 0\}, \mathrm{sigmoid}(z) = 1/(1 + \exp(-z))$
    - $\mathrm{softmax}(z)_i = \exp(z_i)/\sum_j \exp(z_j)$, max pooling, attention, …
  - Usually train via SGD, but it's **non-convex**: in general, possibility of local minima
    - ERM is NP-hard, even with 1 ReLU, *even for square loss* (Goel et al. ITCS 2021)

Universal approximation in $\mathbb{R}$

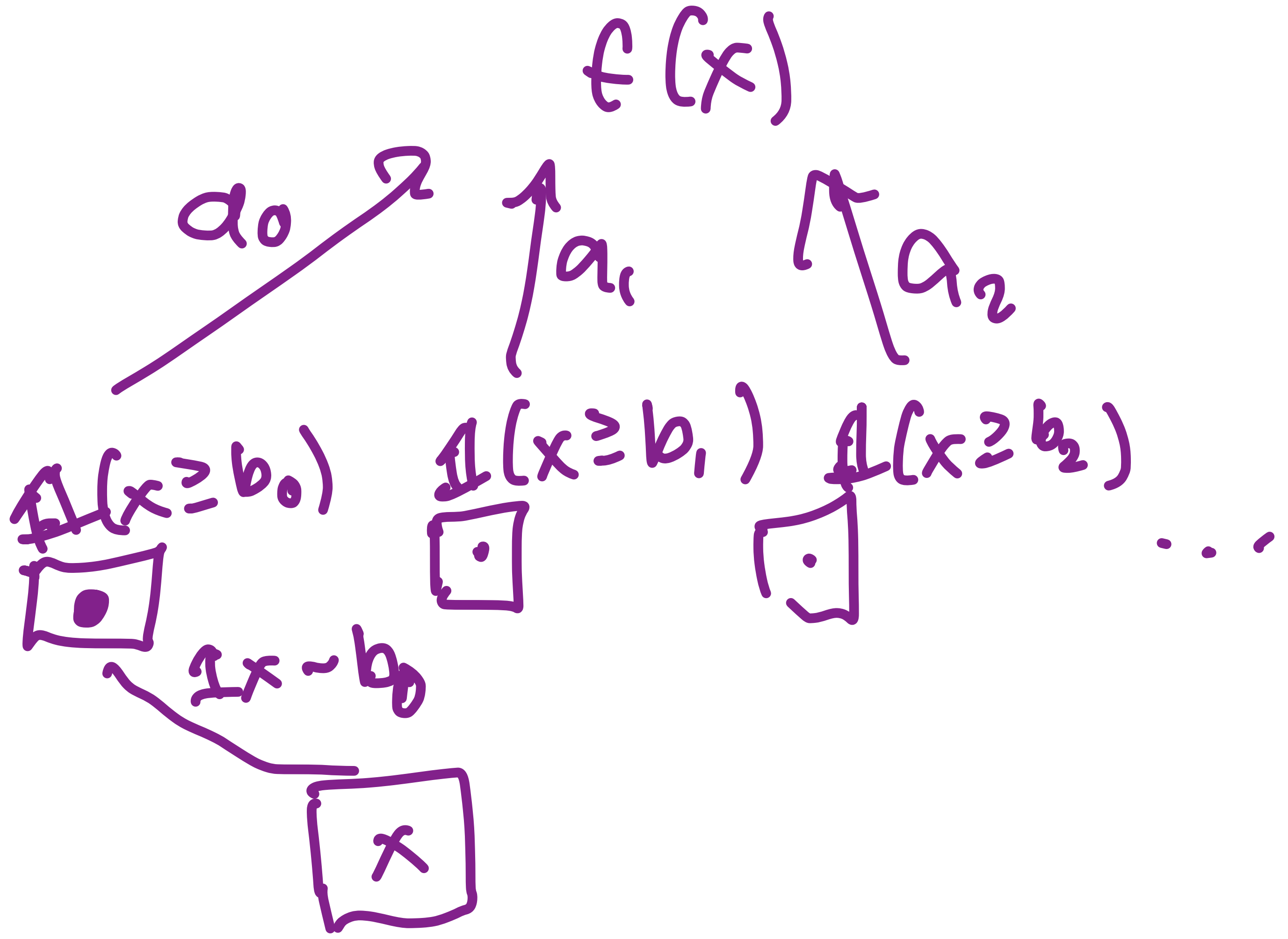
Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{1}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.



Universal approximation in $\mathbb{R}$

Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{1}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.

$$b_i = \frac{i\varepsilon}{\rho} \qquad a_0 = g(0) \qquad a_i = g(b_i) - g(b_{i-1}) \qquad f(x) = \sum_{i=0}^{m-1} a_i \mathbb{1}(x \geq b_i)$$



$$f(b_i) = \sum_{j=0}^{i} a_j = g(b_i) - g(b_{i-1}) + g(b_{i-1}) - \dots - g(0) + g(0)$$

$$= g(b_i)$$

Universal approximation in $\mathbb{R}$

Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{I}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.

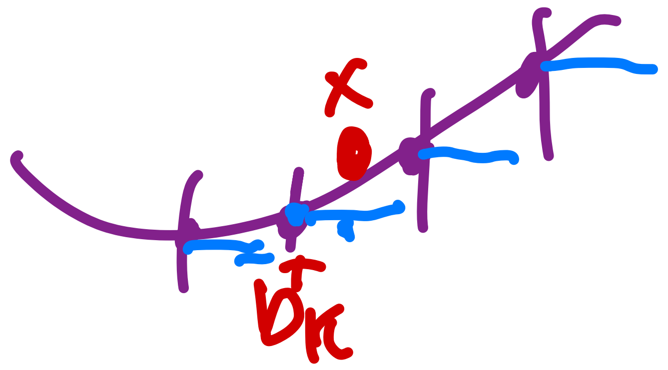
$$b_i = \frac{i\varepsilon}{\rho} \qquad a_0 = g(0) \qquad a_i = g(b_i) - g(b_{i-1}) \qquad f(x) = \sum_{i=0}^{m-1} a_i \mathbb{I}(x_i \geq b_i)$$

$$|\, g(x) - f(x) \,|$$

Universal approximation in $\mathbb{R}$

Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{1}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.

$$b_i = \frac{i\varepsilon}{\rho} \qquad a_0 = g(0) \qquad a_i = g(b_i) - g(b_{i-1}) \qquad f(x) = \sum_{i=0}^{m-1} a_i \mathbb{1}(x_i \geq b_i)$$



$$k = \max\{k : b_k \leq x\}$$

$$|g(x) - f(x)| \leq |g(x) - g(b_k)| + \underbrace{|g(b_k) - f(b_k)|}_{0} + \underbrace{|f(b_k) - f(x)|}_{0}$$

Universal approximation in $\mathbb{R}$

Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{I}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.

$$b_i = \frac{i\varepsilon}{\rho} \qquad a_0 = g(0) \qquad a_i = g(b_i) - g(b_{i-1}) \qquad f(x) = \sum_{i=0}^{m-1} a_i \mathbb{I}(x_i \geq b_i)$$

$$k = \max\{k : b_k \leq x\}$$

$$\begin{aligned} |g(x) - f(x)| &\leq |g(x) - g(b_k)| + |g(b_k) - f(b_k)| + |f(b_k) - f(x)| \\ &\leq \rho |x - b_k| \end{aligned}$$

Universal approximation in $\mathbb{R}$

Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{I}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.

$$b_i = \frac{i\varepsilon}{\rho} \qquad a_0 = g(0) \qquad a_i = g(b_i) - g(b_{i-1}) \qquad f(x) = \sum_{i=0}^{m-1} a_i \mathbb{I}(x_i \geq b_i)$$

$$k = \max\{k : b_k \leq x\}$$

$$\begin{aligned} |g(x) - f(x)| &\leq |g(x) - g(b_k)| + |g(b_k) - f(b_k)| + |f(b_k) - f(x)| \\ &\leq \rho |x - b_k| \\ &\leq \rho \frac{\varepsilon}{\rho} = \varepsilon \end{aligned}$$

Universal approximation in $\mathbb{R}$

Theorem: Let $g : \mathbb{R} \to \mathbb{R}$ be ρ-Lipschitz. For any $\varepsilon > 0$, there is a two-layer network f with $m := \lceil \frac{\rho}{\varepsilon} \rceil$ hidden nodes, $\sigma_1(z) = \mathbb{I}(z \geq 0)$, with $\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$.

$$b_i = \frac{i\varepsilon}{\rho} \qquad a_0 = g(0) \qquad a_i = g(b_i) - g(b_{i-1}) \qquad f(x) = \sum_{i=0}^{m-1} a_i \mathbb{I}(x_i \geq b_i)$$

$$k = \max\{k : b_k \leq x\}$$

$$\begin{aligned} |g(x) - f(x)| &\leq |g(x) - g(b_k)| + |g(b_k) - f(b_k)| + |f(b_k) - f(x)| \\ &\leq \rho |x - b_k| \\ &\leq \rho \frac{\varepsilon}{\rho} = \varepsilon \end{aligned}$$

Can do better by depending on *total variation* of g

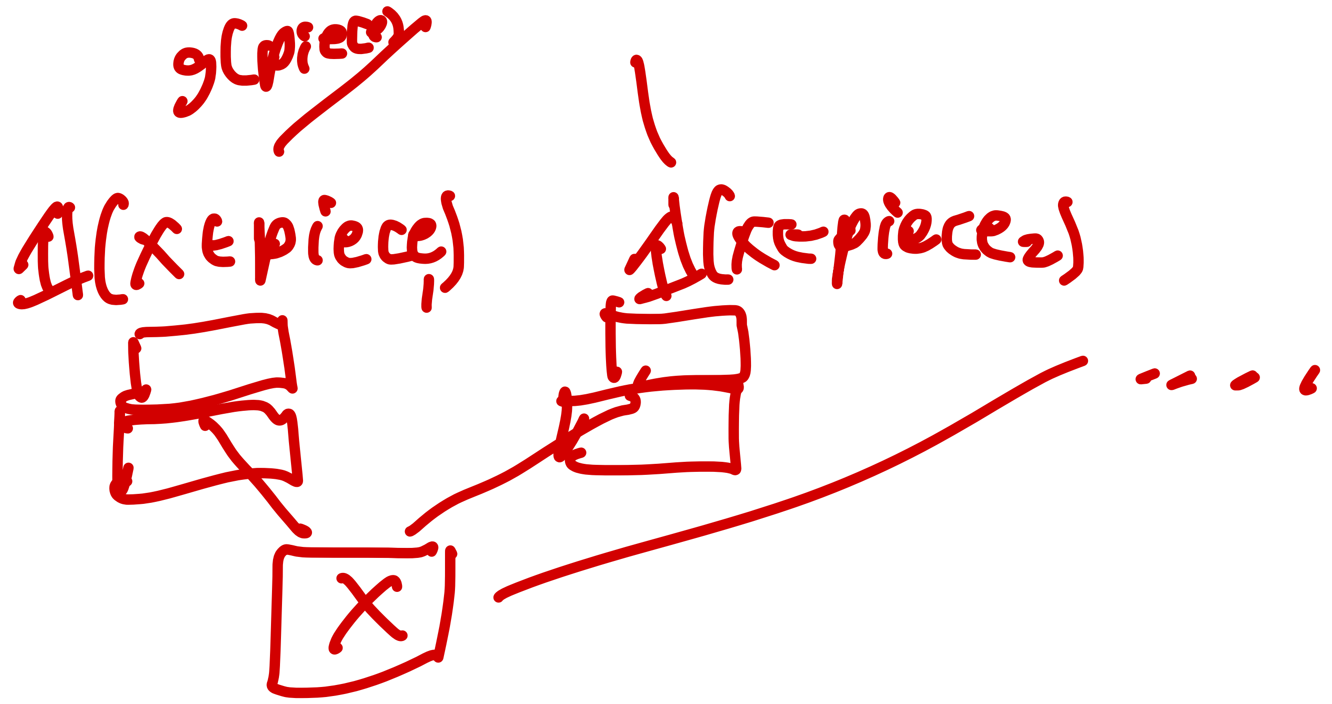
Universal approximation in $\mathbb{R}^d$

Theorem: Let $g : \cancel{\mathbb{R}^d}\ [0,1]^d \to \mathbb{R}$ be continuous. For any $\varepsilon > 0$, choose $\delta > 0$ so that $\|x - x'\|_\infty \leq \delta$ implies $|g(x) - g(x')| \leq \varepsilon$. Then there is a three-layer ReLU network f with $\Omega\left(\frac{1}{\delta^d}\right)$ nodes satisfying $\int_{[0,1]^d} |f(x) - g(x)| \mathrm{d}x \leq 2\varepsilon$.

Universal approximation in $\mathbb{R}^d$

Theorem: Let $g : \mathbb{R}^d \to \mathbb{R}$ be continuous. For any $\varepsilon > 0$, choose $\delta > 0$ so that $\|x - x'\|_\infty \le \delta$ implies $|g(x) - g(x')| \le \varepsilon$. Then there is a three-layer ReLU network f with $\Omega\left(\frac{1}{\delta^d}\right)$ nodes satisfying $\int_{[0,1]^d} |f(x) - g(x)| \mathrm{d}x \le 2\varepsilon$.

Proof approximates continuous g by piecewise-constant h,
then uses a two-layer ReLU net to check if x is in each piece, roughly like in 1d.
(Telgarsky's Theorem 2.1.)



Universal approximation in $\mathbb{R}^d$, one hidden layer

Stone-Weierstrass Theorem: Let $\mathcal{F}$ be a set of functions such that

1. Each $f \in \mathcal{F}$ is continuous.
2. For each x, there is at least one $f \in \mathcal{F}$ with $f(x) \neq 0$.
3. Separates points: for each $x \neq x'$, there is at least one $f \in \mathcal{F}$ with $f(x) \neq f(x')$.
4. $\mathcal{F}$ is an algebra: for $f, g \in \mathcal{F}$, $\quad \alpha f + g \in \mathcal{F} \quad$ and $\quad fg = (x \mapsto f(x)g(x)) \in \mathcal{F}$.

Then $\mathcal{F}$ is L_∞-dense in $C(\mathcal{X})$, i.e. $\forall$ continuous $g : \mathcal{X} \to \mathbb{R}$, $\exists\, f \in \mathcal{F}$ s.t. $\underbrace{\|f-g\|_\infty}_{\sup_{x\in\mathcal{X}} |f(x)-g(x)|} \leq \varepsilon$.

Universal approximation in $\mathbb{R}^d$, one hidden layer

Stone-Weierstrass Theorem: Let $\mathscr{F}$ be a set of functions such that

1. Each $f \in \mathscr{F}$ is continuous.
2. For each x, there is at least one $f \in \mathscr{F}$ with $f(x) \neq 0$.
3. Separates points: for each $x \neq x'$, there is at least one $f \in \mathscr{F}$ with $f(x) \neq f(x')$.
4. $\mathscr{F}$ is an algebra: for $f, g \in \mathscr{F}$, $\quad \alpha f + g \in \mathscr{F}$ and $fg = (x \mapsto f(x)g(x)) \in \mathscr{F}$.

Conditions hold for $\sigma_1 = \exp$, $\sigma_2 = \mathrm{Id}$, so that $\mathscr{F}_{\exp} = \{x \mapsto \sum_{i=1}^{m} a_i \exp(w_i^\top x)\}$

$$\left(\sum_i a_i \exp(w_i^\top x)\right)\left(\sum_j a_j' \exp(w_j'^\top x)\right)$$

$$= \sum_{ij} a_i a_j' \exp((w_i + w_j')^\top x)$$

Universal approximation in $\mathbb{R}^d$, one hidden layer

Stone-Weierstrass Theorem: Let $\mathscr{F}$ be a set of functions such that

1. Each $f \in \mathscr{F}$ is continuous.
2. For each x, there is at least one $f \in \mathscr{F}$ with $f(x) \neq 0$.
3. Separates points: for each $x \neq x'$, there is at least one $f \in \mathscr{F}$ with $f(x) \neq f(x')$.
4. $\mathscr{F}$ is an algebra: for $f, g \in \mathscr{F}$, $\quad \alpha f + g \in \mathscr{F} \quad$ and $\quad fg = (x \mapsto f(x)g(x)) \in \mathscr{F}$.

Conditions hold for $\sigma_1 = \exp$, $\sigma_2 = \mathrm{Id}$, so that $\mathscr{F}_{\exp} = \{x \mapsto \sum_{i=1}^{m} a_i \exp(w_i^\top x)\}$

If $\sigma : \mathbb{R} \to \mathbb{R}$ is continuous, $\lim_{z \to -\infty} \sigma(z) = 0$, $\lim_{z \to -\infty} \sigma(z) = 1$, works too:

Approximate g by $h \in \mathscr{F}_{\exp}$ with $\frac{\varepsilon}{2}$ error, and replace each $\exp$ with a 1d σ-based net

Universal approximation in $\mathbb{R}^d$, one hidden layer

Stone-Weierstrass Theorem: Let $\mathcal{F}$ be a set of functions such that

1. Each $f \in \mathcal{F}$ is continuous.
2. For each x, there is at least one $f \in \mathcal{F}$ with $f(x) \neq 0$.
3. Separates points: for each $x \neq x'$, there is at least one $f \in \mathcal{F}$ with $f(x) \neq f(x')$.
4. $\mathcal{F}$ is an algebra: for $f, g \in \mathcal{F}$, $\quad \alpha f + g \in \mathcal{F} \quad$ and $\quad fg = (x \mapsto f(x)g(x)) \in \mathcal{F}$.

Conditions hold for $\sigma_1 = \exp$, $\sigma_2 = \mathrm{Id}$, so that $\mathcal{F}_{\exp} = \{x \mapsto \sum_{i=1}^{m} a_i \exp(w_i^\top x)\}$

If $\sigma : \mathbb{R} \to \mathbb{R}$ is continuous, $\lim_{z \to -\infty} \sigma(z) = 0$, $\lim_{z \to -\infty} \sigma(z) = 1$, works too:

Approximate g by $h \in \mathcal{F}_{\exp}$ with $\frac{\varepsilon}{2}$ error, and replace each $\exp$ with a 1d σ-based net

Generally: universal approximator iff σ is **not** a polynomial

Limits of universal approximation

- Curse of dimensionality: usually requires # of units exponential in dimension
  - Also usually requires exponential norm of weights

- Doesn’t say anything about whether ERM finds a good network, just that one exists
  - Let alone anything about whether (S)GD finds it

Universal approximation via circuit complexity

SSBD chapter 20:

- 2 layer nets with sign activations can represent all functions $\{\pm 1\}^d \to \{\pm 1\}$

Universal approximation via circuit complexity

SSBD chapter 20:

- 2 layer nets with sign activations can represent all functions $\{\pm 1\}^d \to \{\pm 1\}$
  - (remember that computers always represent things as $\{0,1\}^d$...)

Universal approximation via circuit complexity

SSBD chapter 20:

- 2 layer nets with sign activations can represent all functions $\{\pm 1\}^d \to \{\pm 1\}$
  - (remember that computers always represent things as $\{0,1\}^d$…)
- …but, it takes exponential width to do that

Universal approximation via circuit complexity

SSBD chapter 20:

- 2 layer nets with sign activations can represent all functions $\{\pm 1\}^d \to \{\pm 1\}$
  - (remember that computers always represent things as $\{0,1\}^d$…)
- …but, it takes exponential width to do that
- …but, there's a network of size $\mathcal{O}(T^2)$ that can implement all boolean functions that can be computed in maximum runtime T

Universal approximation via circuit complexity

SSBD chapter 20:

- 2 layer nets with sign activations can represent all functions $\{\pm 1\}^d \to \{\pm 1\}$
    - (remember that computers always represent things as $\{0,1\}^d$…)
- …but, it takes exponential width to do that
- …but, there's a network of size $\mathcal{O}(T^2)$ that can implement all boolean functions that can be computed in maximum runtime T

Universal approximation via circuit complexity

SSBD chapter 20:

- 2 layer nets with sign activations can represent all functions $\{\pm 1\}^d \to \{\pm 1\}$
  - (remember that computers always represent things as $\{0,1\}^d$…)
- …but, it takes exponential width to do that
- …but, there's a network of size $\mathcal{O}(T^2)$ that can implement all boolean functions that can be computed in maximum runtime T

Circuit Complexity and Neural Networks, Ian Parberry (1994) - UBC access

Proceedings of Machine Learning Research vol 125:1–22, 2020 33rd Annual Conference on Learning Theory

Universal Approximation with Deep Narrow Networks

Patrick Kidger KIDGER@MATHS.OX.AC.UK
Terry Lyons TLYONS@MATHS.OX.AC.UK
Mathematical Institute, University of Oxford

Editors: Jacob Abernethy and Shivani Agarwal

Abstract

The classical Universal Approximation Theorem holds for neural networks of arbitrary width and bounded depth. Here we consider the natural 'dual' scenario for networks of bounded width and arbitrary depth. Precisely, let n be the number of inputs neurons, m be the number of output neurons, and let ρ be any nonaffine continuous function, with a continuous nonzero derivative at some point. Then we show that the class of neural networks of arbitrary depth, width $n + m + 2$, and activation function ρ, is dense in $C(K; \mathbb{R}^m)$ for $K \subseteq \mathbb{R}^n$ with K compact. This covers every activation function possible to use in practice, and also includes polynomial activation functions, which is unlike the classical version of the theorem, and provides a qualitative difference between deep narrow networks and shallow wide networks. We then consider several extensions of this result. In particular we consider nowhere differentiable activation functions, density in noncompact domains with respect to the L^p-norm, and how the width may be reduced to just $n + m + 1$ for 'most' activation functions.

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*
  - "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."

    – Y. Bengio & LeCun (2007)

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*
  - "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."
    – Y. Bengio & LeCun (2007)
  - Lots of empirical evidence, but theoretical support pretty limited until recently

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*
  - "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."

    – Y. Bengio & LeCun (2007)
  - Lots of empirical evidence, but theoretical support pretty limited until recently

  - Telgarsky notes section 5 give a particular such function:
    shallow net needs huge width to approximate,
    but narrow not-super-deep net can approximate it efficiently

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*
  - "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."
    – Y. Bengio & LeCun (2007)
  - Lots of empirical evidence, but theoretical support pretty limited until recently

  - Telgarsky notes section 5 give a particular such function:
    shallow net needs huge width to approximate,
    but narrow not-super-deep net can approximate it efficiently
  - Also proved for a certain class of functions by Mhaskar, Liao, Poggio (2016)

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*
  - "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."
    – Y. Bengio & LeCun (2007)
  - Lots of empirical evidence, but theoretical support pretty limited until recently

  - Telgarsky notes section 5 give a particular such function:
    shallow net needs huge width to approximate,
    but narrow not-super-deep net can approximate it efficiently
  - Also proved for a certain class of functions by Mhaskar, Liao, Poggio (2016)
  - Lu et al. (2017): approximating wide nets with deep nets easier(ish) than vice versa

Why deep instead of wide?

- Deep networks much better at learning *compositional structure*
  - "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."
    – Y. Bengio & LeCun (2007)
  - Lots of empirical evidence, but theoretical support pretty limited until recently

  - Telgarsky notes section 5 give a particular such function:
    shallow net needs huge width to approximate,
    but narrow not-super-deep net can approximate it efficiently
  - Also proved for a certain class of functions by Mhaskar, Liao, Poggio (2016)
  - Lu et al. (2017): approximating wide nets with deep nets easier(ish) than vice versa
  - Liang and Srikant (2017): can approximate piecewise-constant funcs with exponentially smaller deep nets than shallow

- We have some universal approximation results
- A lot of people use this to say “neural networks can do anything! 💪”

- We have some universal approximation results
- A lot of people use this to say “neural networks can do anything! 💪”

but…

- We have some universal approximation results
- A lot of people use this to say “neural networks can do anything! 💪”

but…

- These kind of approximation results don’t tell us:
    - What practically-sized networks can do

- We have some universal approximation results
- A lot of people use this to say "neural networks can do anything! 💪"

but…

- These kind of approximation results don't tell us:
  - What practically-sized networks can do
    - Gaussian kernels can also do anything (💪)…with ridiculously large norm

$K(x,y) = \psi(x-y)$ is universal iff $\mathcal{F}[\psi] > 0$ everywhere

$= \sum_{k=0}^{\infty} a_k \langle x,y \rangle^k$ " iff $\forall k,\ a_k > 0$

- We have some universal approximation results
- A lot of people use this to say “neural networks can do anything! 💪”

but…

- These kind of approximation results don’t tell us:
  - What practically-sized networks can do
    - Gaussian kernels can also do anything (💪)…with ridiculously large norm
    - Neural nets can do anything…if they’re ridiculously large (or large norm)

- We have some universal approximation results
- A lot of people use this to say “neural networks can do anything! 💪”

but…

- These kind of approximation results don’t tell us:
  - What practically-sized networks can do
    - Gaussian kernels can also do anything (💪)…with ridiculously large norm
    - Neural nets can do anything…if they’re ridiculously large (or large norm)

  - Even if our class approximates, do we generalize? (Does ERM, RLM, … work?)

- We have some universal approximation results
- A lot of people use this to say “neural networks can do anything! 💪”

but…

- These kind of approximation results don’t tell us:
  - What practically-sized networks can do
    - Gaussian kernels can also do anything (💪)…with ridiculously large norm
    - Neural nets can do anything…if they’re ridiculously large (or large norm)
  - Even if our class approximates, do we generalize? (Does ERM, RLM, … work?)
  - Does (S)GD find an approximate ERM / RLM / something that generalizes?

- We have some universal approximation results
- A lot of people use this to say "neural networks can do anything! 💪"

but…

- These kind of approximation results don't tell us:
  - What practically-sized networks can do
    - Gaussian kernels can also do anything (💪)…with ridiculously large norm
    - Neural nets can do anything…if they're ridiculously large (or large norm)
  - Even if our class approximates, do we generalize? (Does ERM, RLM, … work?)
  - Does (S)GD find an approximate ERM / RLM / something that generalizes?
    - We (pretty much) know it doesn't *always* find an (approximate) ERM: ERM with deep nets (even for square loss) is NP-hard

so, if you can prove that it *does*, let me know =)

Generalization: VC dimension

- For ReLU (or general piecewise-linear) nets with P params, VCdim = $\mathcal{O}(PL \log P)$
  - and $\Omega\left(PL \log \frac{P}{L}\right)$, so nearly tight – Bartlett/Harvey/Liaw/Mehrabian (2019)

Generalization: VC dimension

- For ReLU (or general piecewise-linear) nets with P params, VCdim = $\mathcal{O}(PL \log P)$
  - and $\Omega\left(PL \log \frac{P}{L}\right)$, so nearly tight – Bartlett/Harvey/Liaw/Mehrabian (2019)
- $P = \prod_{\ell=1}^{L} d_{\ell-1} d_\ell$ for fully-connected networks

Generalization: VC dimension

- For ReLU (or general piecewise-linear) nets with P params, VCdim = $\mathcal{O}(PL \log P)$
  - and $\Omega\left(PL \log \frac{P}{L}\right)$, so nearly tight – Bartlett/Harvey/Liaw/Mehrabian (2019)
  - $P = \prod_{\ell=1}^{L} d_{\ell-1} d_\ell$ for fully-connected networks
- For piecewise-constant, e.g. threshold functions, VCdim = $\Theta(P \log P)$

Generalization: VC dimension

- For ReLU (or general piecewise-linear) nets with P params, VCdim = $\mathcal{O}(PL \log P)$
  - and $\Omega\left(PL \log \frac{P}{L}\right)$, so nearly tight – Bartlett/Harvey/Liaw/Mehrabian (2019)
  - $P = \prod_{\ell=1}^{L} d_{\ell-1} d_\ell$ for fully-connected networks
- For piecewise-constant, e.g. threshold functions, VCdim = $\Theta(P \log P)$
- For piecewise-polynomial, $\mathcal{O}(PL^2 + PL \log P)$, $\mathcal{O}(PU)$ with U units

Generalization: VC dimension

- For ReLU (or general piecewise-linear) nets with P params, VCdim = $\mathcal{O}(PL \log P)$
  - and $\Omega\left(PL \log \frac{P}{L}\right)$, so nearly tight – Bartlett/Harvey/Liaw/Mehrabian (2019)
  - $P = \prod_{\ell=1}^{L} d_{\ell-1} d_\ell$ for fully-connected networks
- For piecewise-constant, e.g. threshold functions, VCdim = $\Theta(P \log P)$
- For piecewise-polynomial, $\mathcal{O}(PL^2 + PL \log P)$, $\mathcal{O}(PU)$ with U units
- For sigmoids/similar, $\mathcal{O}(P^2 U^2)$ and $\Omega(P^2)$

Generalization: VC dimension

- For ReLU (or general piecewise-linear) nets with P params, VCdim = $\mathcal{O}(PL \log P)$
  - and $\Omega\left(PL \log \frac{P}{L}\right)$, so nearly tight – Bartlett/Harvey/Liaw/Mehrabian (2019)
  - $P = \prod_{\ell=1}^{L} d_{\ell-1} d_\ell$ for fully-connected networks
- For piecewise-constant, e.g. threshold functions, VCdim = $\Theta(P \log P)$
- For piecewise-polynomial, $\mathcal{O}(PL^2 + PL \log P)$, $\mathcal{O}(PU)$ with U units
- For sigmoids/similar, $\mathcal{O}(P^2 U^2)$ and $\Omega(P^2)$
  - Theorem 8.13/8.14 of Anthony & Bartlett (1999) textbook - UBC access

Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion

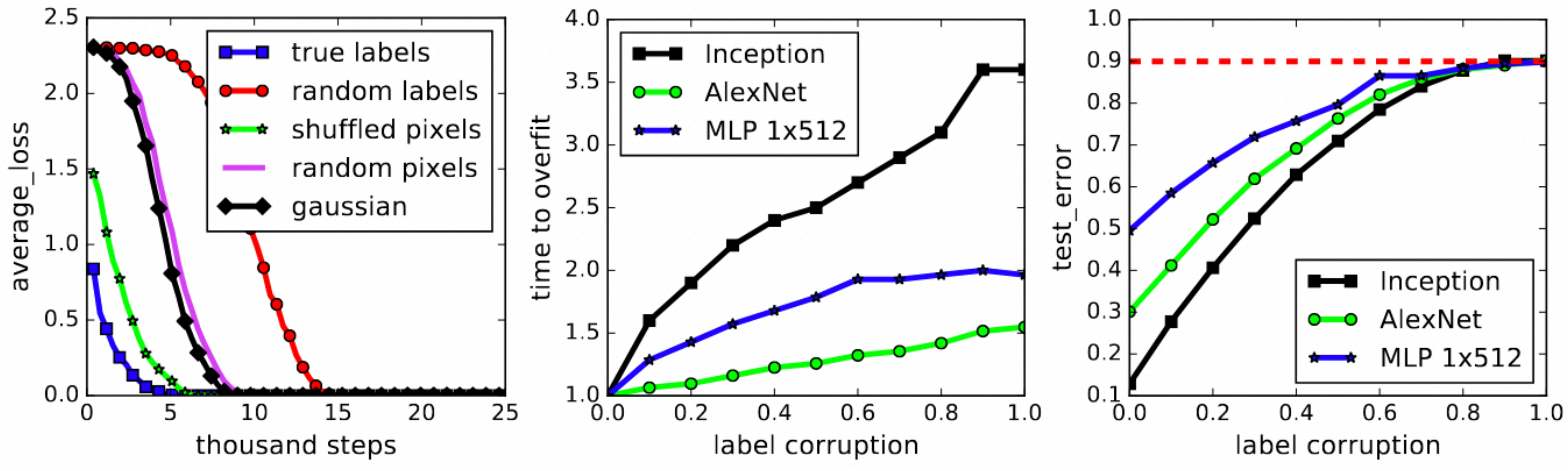
Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion
- We can train our networks to get zero error even for **random labels**

Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion

- We can train our networks to get zero error even for **random labels**
  - Even AlexNet can shatter CIFAR-10, *almost* shatter ImageNet
  - <u>Neyshabur et al. (2015)</u>, <u>Zhang et al. (2017)</u>



(a) learning curves (b) convergence slowdown (c) generalization error growth

Figure 1: Fitting random labels and random pixels on CIFAR10. (a) shows the training loss of various experiment settings decaying with the training steps. (b) shows the relative convergence time with different label corruption ratio. (c) shows the test error (also the generalization error since training error is 0) under different label corruptions.

Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion
- We can train our networks to get zero error even for **random labels**
  - Even AlexNet can shatter CIFAR-10, *almost* shatter ImageNet
  - Neyshabur et al. (2015), Zhang et al. (2017)
  - But these architectures do generalize well – VC of arch. can't explain that

Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion

- We can train our networks to get zero error even for **random labels**
  - Even AlexNet can shatter CIFAR-10, *almost* shatter ImageNet
  - Neyshabur et al. (2015), Zhang et al. (2017)
  - But these architectures do generalize well – VC of arch. can't explain that
  - Uniform stability can't either, since it's data-independent;
    on-average replace-one stability always can, but hard

Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion

- We can train our networks to get zero error even for **random labels**
  - Even AlexNet can shatter CIFAR-10, *almost* shatter ImageNet
  - Neyshabur et al. (2015), Zhang et al. (2017)
  - But these architectures do generalize well – VC of arch. can't explain that
  - Uniform stability can't either, since it's data-independent;
    on-average replace-one stability always can, but hard

- Making hidden layers wider can often improve generalization,
  but worsens parameter counting-based bounds

Problems with parameter counting

- We use networks with a **lot** of parameters
  - ResNet-50 has ~25 million parameters and depth 50: VCdim > 1 billion

- We can train our networks to get zero error even for **random labels**
  - Even AlexNet can shatter CIFAR-10, *almost* shatter ImageNet
  - Neyshabur et al. (2015), Zhang et al. (2017)
  - But these architectures do generalize well – VC of arch. can't explain that
  - Uniform stability can't either, since it's data-independent;
    on-average replace-one stability always can, but hard

- Making hidden layers wider can often improve generalization,
  but worsens parameter counting-based bounds
- Remember that $\cancel{\mathcal{H}_k}$ $\mathcal{F}$ has infinite VCdim for universal kernels,
  but we can still learn with *small-norm* predictors

Theorem: Fix $\sigma_1, \dots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \dots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\|\left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b\right)\right\|_c$$

Base case, $L = 0$:

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \leq B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \leq \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\|\left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b\right)\right\|_c$$

Base case, $L = 0$:

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\})$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$:

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) \le \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2 \right)\sqrt{2\log d}$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\|\left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b\right)\right\|_c$$

Base case, $L = 0$:

$$\begin{aligned}\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) &\le \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2\right)\sqrt{2\log d} \\ &= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d}\end{aligned}$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\|\left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b\right)\right\|_c$$

Base case, $L = 0$:

$$\begin{aligned}\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) &\le \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2\right)\sqrt{2\log d} \\ &= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}\end{aligned}$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \leq B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \leq \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$:

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) \leq \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2 \right)\sqrt{2\log d}$$
$$= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}$$

Inductive step:

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$:

$$\begin{aligned}\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) &\le \frac{1}{n}\Big(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2 \Big)\sqrt{2\log d} \\ &= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}\end{aligned}$$

Inductive step:

$$\hat{\mathfrak{R}}_S(\mathscr{F}_{\ell+1}) = \hat{\mathfrak{R}}_S\left(\left\{ x \mapsto \sigma_{\ell+1}\left(\|W_{\ell+1}^\top\|_{1,\infty}\, g(x) \right) : g \in \operatorname{conv}\big(-\mathscr{F}_\ell \cup \mathscr{F}_\ell \big) \right\} \right)$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$**:**

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) \le \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2 \right)\sqrt{2\log d}$$
$$= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}$$

Inductive step:

$$\hat{\mathfrak{R}}_S(\mathscr{F}_{\ell+1}) = \hat{\mathfrak{R}}_S\left(\left\{ x \mapsto \sigma_{\ell+1}\left(\|W_{\ell+1}^\mathsf{T}\|_{1,\infty}\, g(x) \right) : g \in \operatorname{conv}\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell \right) \right\} \right)$$
$$\le \rho B\, \hat{\mathfrak{R}}_S\left(\operatorname{conv}(-\mathscr{F}_\ell \cup \mathscr{F}_\ell) \right)$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\|\left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b\right)\right\|_c$$

Base case, $L = 0$:

$$\begin{aligned}\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) &\le \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2\right)\sqrt{2\log d}\\ &= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}\end{aligned}$$

Inductive step:

$$\begin{aligned}\hat{\mathfrak{R}}_S(\mathscr{F}_{\ell+1}) &= \hat{\mathfrak{R}}_S\left(\left\{x \mapsto \sigma_{\ell+1}\left(\|W_{\ell+1}^\top\|_{1,\infty}\, g(x)\right) : g \in \operatorname{conv}\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell\right)\right\}\right)\\ &\le \rho B\, \hat{\mathfrak{R}}_S\left(\operatorname{conv}(-\mathscr{F}_\ell \cup \mathscr{F}_\ell)\right)\end{aligned}$$

$$\hat{\mathfrak{R}}_S(\operatorname{conv}(\mathscr{G})) = \hat{\mathfrak{R}}_S(\mathscr{G})$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \leq B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \leq \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$:

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) \leq \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2 \right)\sqrt{2\log d}$$
$$= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}$$

Inductive step:

$$\hat{\mathfrak{R}}_S(\mathscr{F}_{\ell+1}) = \hat{\mathfrak{R}}_S\left(\left\{ x \mapsto \sigma_{\ell+1}\left(\|W_{\ell+1}^\top\|_{1,\infty}\, g(x) \right) : g \in \operatorname{conv}\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell \right) \right\} \right)$$
$$\leq \rho B\, \hat{\mathfrak{R}}_S\left(\operatorname{conv}(-\mathscr{F}_\ell \cup \mathscr{F}_\ell) \right) \quad \hat{\mathfrak{R}}_S(\operatorname{conv}(\mathscr{G})) = \hat{\mathfrak{R}}_S(\mathscr{G})$$
$$\leq \rho B\, \hat{\mathfrak{R}}_S\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell \right)$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$:

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) \le \frac{1}{n}\left(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2 \right)\sqrt{2\log d}$$
$$= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}$$

Inductive step:

$$\hat{\mathfrak{R}}_S(\mathscr{F}_{\ell+1}) = \hat{\mathfrak{R}}_S\left(\left\{ x \mapsto \sigma_{\ell+1}\left(\|W_{\ell+1}^\top\|_{1,\infty}\, g(x) \right) : g \in \operatorname{conv}\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell \right) \right\} \right)$$
$$\le \rho B\, \hat{\mathfrak{R}}_S\left(\operatorname{conv}(-\mathscr{F}_\ell \cup \mathscr{F}_\ell) \right) \qquad \hat{\mathfrak{R}}_S(\operatorname{conv}(\mathscr{G})) = \hat{\mathfrak{R}}_S(\mathscr{G})$$
$$\le \rho B\, \hat{\mathfrak{R}}_S\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell \right)$$

$\hat{\mathfrak{R}}_S(A \cup B) \le \hat{\mathfrak{R}}_S(A) + \hat{\mathfrak{R}}_S(B)$ if $0 \in A, 0 \in B$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\top\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

Rad($\mathcal{F}|_S$)

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Base case, $L = 0$:

$$\hat{\mathfrak{R}}_S(\{x \mapsto x_j : j \in [d]\}) \le \frac{1}{n}\Big(\max_j \|(x_{1,j}, \ldots, x_{n,j})\|_2\Big)\sqrt{2\log d}$$
$$= \frac{1}{n}\|X\|_{2,\infty}\sqrt{2\log d} = \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^0\sqrt{2\log d}$$

Inductive step:

$$\hat{\mathfrak{R}}_S(\mathscr{F}_{\ell+1}) = \hat{\mathfrak{R}}_S\left(\left\{x \mapsto \sigma_{\ell+1}\left(\|W_{\ell+1}^\top\|_{1,\infty}\, g(x)\right) : g \in \mathrm{conv}\big(-\mathscr{F}_\ell \cup \mathscr{F}_\ell\big)\right\}\right)$$
$$\le \rho B\, \hat{\mathfrak{R}}_S\left(\mathrm{conv}(-\mathscr{F}_\ell \cup \mathscr{F}_\ell)\right) \qquad \hat{\mathfrak{R}}_S(\mathrm{conv}(\mathscr{G})) = \hat{\mathfrak{R}}_S(\mathscr{G})$$
$$\le \rho B\, \hat{\mathfrak{R}}_S\left(-\mathscr{F}_\ell \cup \mathscr{F}_\ell\right) \qquad \hat{\mathfrak{R}}_S(A \cup B) \le \hat{\mathfrak{R}}_S(A) + \hat{\mathfrak{R}}_S(B) \text{ if } 0 \in A, 0 \in B$$
$$\le 2\rho B\, \hat{\mathfrak{R}}_S\left(\mathscr{F}_\ell\right)$$

Rademacher of convex hull

$$\hat{\mathfrak{R}}_S(\mathrm{conv}(\mathcal{G})) = \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha \in \Delta_k} \sup_{g_1,\ldots,g_k \in \mathcal{G}} \left\langle \sigma, \sum_{j=1}^{k} \alpha_j (g_j)_S \right\rangle$$

Rademacher of convex hull

$$\hat{\mathfrak{R}}_S(\mathrm{conv}(\mathcal{G})) = \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sup_{g_1,\ldots,g_k\in\mathcal{G}} \left\langle \sigma, \sum_{j=1}^{k} \alpha_j (g_j)_S \right\rangle$$

$$= \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sum_{j=1}^{k} \alpha_j \sup_{g_j\in\mathcal{G}} \left\langle \varepsilon, (g_j)_S \right\rangle$$

Rademacher of convex hull

$$\hat{\mathfrak{R}}_S(\mathrm{conv}(\mathscr{G})) = \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sup_{g_1,\ldots,g_k\in\mathscr{G}} \left\langle \sigma, \sum_{j=1}^{k} \alpha_j (g_j)_S \right\rangle$$

$$= \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sum_{j=1}^{k} \alpha_j \sup_{g_j\in\mathscr{G}} \left\langle \varepsilon, (g_j)_S \right\rangle$$

$$= \frac{1}{n}\mathbb{E}_\sigma \left(\sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sum_{j=1}^{k} \alpha_j \right) \sup_{g\in\mathscr{G}} \left\langle \varepsilon, g_S \right\rangle$$

Rademacher of convex hull

$$\hat{\mathfrak{R}}_S(\mathrm{conv}(\mathscr{G})) = \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sup_{g_1,\ldots,g_k\in\mathscr{G}} \left\langle \sigma, \sum_{j=1}^{k} \alpha_j (g_j)_S \right\rangle$$

$$= \frac{1}{n}\mathbb{E}_\sigma \sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sum_{j=1}^{k} \alpha_j \sup_{g_j\in\mathscr{G}} \left\langle \varepsilon, (g_j)_S \right\rangle$$

$$= \frac{1}{n}\mathbb{E}_\sigma \left(\sup_{k\geq 1} \sup_{\alpha\in\Delta_k} \sum_{j=1}^{k} \alpha_j \right) \sup_{g\in\mathscr{G}} \left\langle \varepsilon, g_S \right\rangle$$

$$= \hat{\mathfrak{R}}_S(\mathscr{G})$$

Rademacher of union

$$\hat{\mathfrak{R}}_S(\mathscr{G} \cup \mathscr{H}) = \frac{1}{n}\mathbb{E}_\sigma \sup_{g \in (\mathscr{G} \cup \mathscr{H})} \langle \sigma, g_S \rangle$$

Rademacher of union

$$\hat{\mathfrak{R}}_S(\mathcal{G} \cup \mathcal{H}) = \frac{1}{n}\mathbb{E}_\sigma \sup_{g\in(\mathcal{G}\cup\mathcal{H})} \left\langle \sigma, g_S \right\rangle$$

$$\leq \frac{1}{n}\mathbb{E}_\sigma\Big[\sup_{g\in\mathcal{G}} \left\langle \sigma, g_S \right\rangle + \sup_{g\in\mathcal{H}} \left\langle \sigma, g_S \right\rangle\Big] \quad \text{if } 0 \in \mathcal{G}, 0 \in \mathcal{H}$$

Rademacher of union

$$\hat{\mathfrak{R}}_S(\mathscr{G} \cup \mathscr{H}) = \frac{1}{n}\mathbb{E}_\sigma \sup_{g \in (\mathscr{G} \cup \mathscr{H})} \langle \sigma, g_S \rangle$$

$$\leq \frac{1}{n}\mathbb{E}_\sigma \Big[\sup_{g \in \mathscr{G}} \langle \sigma, g_S \rangle + \sup_{g \in \mathscr{H}} \langle \sigma, g_S \rangle \Big] \quad \text{if } 0 \in \mathscr{G}, 0 \in \mathscr{H}$$

$$= \hat{\mathfrak{R}}_S(\mathscr{G}) + \hat{\mathfrak{R}}_S(\mathscr{H})$$

Rademacher of union

$$\hat{\mathfrak{R}}_S(\mathcal{G} \cup \mathcal{H}) = \frac{1}{n}\mathbb{E}_\sigma \sup_{g \in (\mathcal{G} \cup \mathcal{H})} \langle \sigma, g_S \rangle$$

$$\leq \frac{1}{n}\mathbb{E}_\sigma \Big[\sup_{g \in \mathcal{G}} \langle \sigma, g_S \rangle + \sup_{g \in \mathcal{H}} \langle \sigma, g_S \rangle \Big] \quad \text{if } 0 \in \mathcal{G}, 0 \in \mathcal{H}$$

or if both sets are **symmetric**:

for all $g \in \mathcal{G}$, also have $-g \in \mathcal{G}$

$$= \hat{\mathfrak{R}}_S(\mathcal{G}) + \hat{\mathfrak{R}}_S(\mathcal{H})$$

Rademacher of union

$$\hat{\mathfrak{R}}_S(\mathscr{G} \cup \mathscr{H}) = \frac{1}{n}\mathbb{E}_\sigma \sup_{g \in (\mathscr{G} \cup \mathscr{H})} \langle \sigma, g_S \rangle$$

$$\leq \frac{1}{n}\mathbb{E}_\sigma\Big[\sup_{g \in \mathscr{G}} \langle \sigma, g_S \rangle + \sup_{g \in \mathscr{H}} \langle \sigma, g_S \rangle\Big] \quad \text{if } 0 \in \mathscr{G}, 0 \in \mathscr{H}$$

$$= \hat{\mathfrak{R}}_S(\mathscr{G}) + \hat{\mathfrak{R}}_S(\mathscr{H})$$

or if both sets are **symmetric**:

for all $g \in \mathscr{G}$, also have $-g \in \mathscr{G}$

…or if we otherwise know that

$$\sup_{g \in \mathscr{G}} \langle \sigma, g_S \rangle \geq 0, \quad \sup_{g \in \mathscr{H}} \langle \sigma, g_S \rangle \geq 0$$

for **any** assignment of σ

Rademacher of deep nets

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \leq B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \leq \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Rademacher of deep nets

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \leq B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \leq \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each 1-Lipschitz, positive homogenous ($\sigma_\ell(ax) = a\sigma_\ell(x)$ for $a > 0$).
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell\|_F \leq B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \leq \frac{1}{n}\|X\|_F B^L \left(1 + \sqrt{2L\log 2}\right)$.

(More complicated proof: Golowich/Rakhlin/Shamir, COLT 2018 / Telgarsky's 14.2.)

Rademacher of deep nets

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each ρ-Lipschitz with $\sigma_\ell(0) = 0$.
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell^\mathsf{T}\|_{1,\infty} \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_{2,\infty}(2\rho B)^L\sqrt{2\log d}$.

$$\|M\|_{b,c} = \left\| \left(\|M_{\cdot 1}\|_b, \ldots, \|M_{\cdot d}\|_b \right) \right\|_c$$

Theorem: Fix $\sigma_1, \ldots, \sigma_L$ each 1-Lipschitz, positive homogenous ($\sigma_\ell(ax) = a\sigma_\ell(x)$ for $a > 0$).
Let $\mathscr{F}_L$ be the set of L-layer no-intercept nets, $f^{(\ell)} = \sigma_\ell(W_\ell f^{(\ell-1)})$,
with $\|W_\ell\|_F \le B$. Then $\hat{\mathfrak{R}}_n(\mathscr{F}) \le \frac{1}{n}\|X\|_F B^L \left(1 + \sqrt{2L\log 2}\right)$.

(More complicated proof: Golowich/Rakhlin/Shamir, COLT 2018 / Telgarsky's 14.2.)

Can get a slightly better rate via covering numbers: see Telgarsky's section 16.2.

So, does this solve it?

- Experiment by Dziugaite/Roy (2017):
  training a small network on MNIST (0-4 vs 5-9),
  plotting a Rademacher-based margin bound
  using a different (but similarly[?] tight)
  upper bound on the Rademacher complexity