Approximation for Deep Learning: Going Deep

김지수 (Jisu KIM)

딥러닝의 통계적 이해 (Deep Learning: Statistical Perspective), 2024년 2학기

This lecture note is a combination of Prof. Joong-Ho Won's "Deep Learning: Statistical Perspective" with other lecture notes. Main references are:

Tong Zhang, Mathematical Analysis of Machine Learning Algorithms, https://tongzhang-ml.org/lt-book.html

Matus Telgarsky, Deep learning theory lecture notes, https://mjt.cs.illinois.edu/dlt/

Weinan E, Chao Ma, Stephan Wojtowytsch, Lei Wu, Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't, https://arxiv.org/abs/2009.10713/

Antonio Álvarez López, Breaking the curse of dimensionality with Barron spaces, https://dcn.nat.fau.eu/breaking-the-curse-of-dimensionality-with-barron-spaces/

1 Review

1.1 Basic Model for Supervised Learning

- Input(입력) / Covariate(설명 변수) : $x \in \mathbb{R}^d$, so $x = (x_1, \ldots, x_d)$.
- Output(출력) / Response(반응 변수): $y \in \mathcal{Y}$. If y is categorical, then supervised learning is "classification", and if y is continuous, then supervised learning is "regression".
- Model(모형) :
$$y \approx f(x).$$
If we include the error ϵ to the model, then it can be also written as
$$y = \phi(f(x), \epsilon).$$
For many cases, we assume additive noise, so
$$y = f(x) + \epsilon.$$
- Assumption(가정): f belongs to a family of functions $\mathcal{M}$. This is the assumption of a model: a model can be still used when the corresponding assumption is not satisfied in your data.
- Loss function(손실 함수): $\ell(y, a)$. A loss function measures the difference between estimated and true values for an instance of data.
- Training data(학습 자료): $\mathcal{T} = \{(y_i, x_i), i = 1, \ldots, n\}$, where (y_i, x_i) is a sample from a probability distribution P_i. For many cases we assume i.i.d., or x_i's are fixed and y_i's are i.i.d..
- Goal(목적): we want to find f that minimizes the expected prediction error,
$$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P}\left[\ell(Y, f(X))\right].$$
Here, $\mathcal{F}$ can be different from $\mathcal{M}$; $\mathcal{F}$ can be smaller then $\mathcal{M}$.
- Prediction model(예측 모형): f^0 is unknown, so we estimate f^0 by $\hat{f}$ using data. For many cases we minimizes on the empirical prediction error, that is taking the expectation on the empirical distribution $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{(Y_i, X_i)}$.
$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{P_n}\left[\ell(Y, f(X))\right] = \arg\min_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \ell(Y_i, f(X_i)).$$
- Prediction(예측): if $\hat{f}$ is a predicted function, and x is a new input, then we predict unknown y by $\hat{f}(x)$.

1.2 Hölder Spaces and Sobolev Spaces

The class of Lipschitz functions $H(1, L)$ on $T \subset \mathbb{R}$ is the set of functions $g : T \to \mathbb{R}$ such that

$$|g(y) - g(x)| \leq L|x - y| \text{ for all } x, y \in T.$$

A differentiable function is Lipschitz if and only if it has bounded derivative. Conversely a Lipschitz function is differentiable almost everywhere.

Let $T \subset \mathbb{R}$ and let β be an integer. The Hölder space $H(\beta, L)$ is the set of functions $g : T \to \mathbb{R}$ such that g is $\ell = \beta - 1$ times differentiable and satisfies

$$|g^{(\ell)}(y) - g^{(\ell)}(x)| \leq L|x - y| \text{ for all } x, y \in T.$$

(There is an extension to real valued β but we will not need that.) If $g \in H(\beta, L)$ and $\ell = \beta - 1$, then we can define the Taylor approximation of g at x by

$$\tilde{g}(y) = g(y) + (y - x)g'(x) + \cdots + \frac{(y - x)^\ell}{\ell!} g^{(\ell)}(x)$$

and then $|g(y) - \tilde{g}(y)| \leq |y - x|^\beta$.

The definition for higher dimensions is similar. Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$. Let β and L be positive numbers. Given a vector $s = (s_1, \ldots, s_d)$, define $|s| = s_1 + \cdots + s_d$, $s! = s_1! \cdots s_d!$, $x^s = x_1^{s_1} \cdots x_d^{s_d}$ and

$$D^s = \frac{\partial^{s_1 + \cdots + s_d}}{\partial x_1^{s_1} \cdots \partial x_d^{s_d}}.$$

Let β be a positive integer. Define the *Hölder class*

$$H_d(\beta, L) = \left\{ g : \ |D^s g(x) - D^s g(y)| \leq L\|x - y\|, \quad \text{for all } s \text{ such that } |s| = \beta - 1, \text{ and all } x, y \right\}.$$

For example, if $d = 1$ and $\beta = 2$ this means that

$$|g'(x) - g'(y)| \leq L\,|x - y|, \quad \text{for all } x, y.$$

The most common case is $\beta = 2$; roughly speaking, this means that the functions have bounded second derivatives.

Again, if $g \in H_d(\beta, L)$ then $g(x)$ is close to its Taylor series approximation:

$$|g(u) - g_{x,\beta}(u)| \leq L\|u - x\|^\beta \tag{1}$$

where

$$g_{x,\beta}(u) = \sum_{|s| \leq \beta} \frac{(u - x)^s}{s!} D^s g(x).$$

In the common case of $\beta = 2$, this means that

$$\left| p(u) - [p(x) + (x - u)^T \nabla p(x)] \right| \leq L||x - u||^2.$$

The Sobolev class $S_1(\beta, L)$ on $T \subset \mathbb{R}$ is the set of β times differentiable functions (technically, it only requires weak derivatives) $g : T \to \mathbb{R}$ such that

$$\int_T (g^{(\beta)}(x))^2 dx \leq L^2.$$

Again this extends naturally to $\mathbb{R}^d$. Also, there is an extension to non-integer β. It can be shown that if T is bounded, then $H_d(\beta, L) \subset S_d(\beta, L')$ for appropriate L and L'.

1.3 Rademacher complexity

Random variables $\sigma_1, \ldots, \sigma_n$ are called *Rademacher random variables* if they are independent, identically distributed and $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$. Define the *Rademacher complexity* of $\mathcal{F}$ by

$$\mathsf{Rad}_n(\mathcal{F}) = \mathbb{E}\left(\sup_{f\in\mathcal{F}}\left(\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(Z_i)\right)\right).$$

Some authors use a slightly different definition, namely,

$$\mathsf{Rad}_n(\mathcal{F}) = \mathbb{E}\left(\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(Z_i)\right|\right).$$

You can use either one. They lead to essentially the same results.

Intuitively, $\mathsf{Rad}_n(\mathcal{F})$ is large if we can find functions $f \in \mathcal{F}$ that "look like" random noise, that is, they are highly correlated with $\sigma_1, \ldots, \sigma_n$. Here are some properties of the Rademacher complexity.

Lemma. 1. *If* $\mathcal{F} \subset \mathcal{G}$ *then* $\mathsf{Rad}_n(\mathcal{F}, Z^n) \leq \mathsf{Rad}_n(\mathcal{G}, Z^n)$.

2. *Let* $\mathsf{conv}(\mathcal{F})$ *denote the convex hull of* $\mathcal{F}$. *Then* $\mathsf{Rad}_n(\mathcal{F}, Z^n) = \mathsf{Rad}_n(\mathsf{conv}(\mathcal{F}), Z^n)$.

3. *For any* $c \in \mathbb{R}$, $\mathsf{Rad}_n(c\mathcal{F}, Z^n) = |c|\mathsf{Rad}_n(\mathcal{F}, Z^n)$.

4. *Let* $g : \mathbb{R} \to \mathbb{R}$ *be such that* $g(0) = 0$ *and,* $|g(y) - g(x)| \leq L|x - y|$ *for all* x, y. *Then* $\mathsf{Rad}_n(g \circ \mathcal{F}, Z^n) \leq 2L\mathsf{Rad}_n(\mathcal{F}, Z^n)$.

1.4 Two Layer Neural Networks

A two-layer neural network takes an input vector of d variables $x = (x_1, x_2, \ldots, x_d)$ and builds a nonlinear function $f(x)$ to predict the response $y \in \mathbb{R}^D$. What distinguishes neural networks from other nonlinear methods is the particular structure of the model:

$$f(x) = f_\theta(x) = g\left(\beta_0 + \sum_{j=1}^{m}\beta_j\sigma(b_j + w_j^\top x)\right),$$

where $x \in \mathbb{R}^d, b_j \in \mathbb{R}, w_j \in \mathbb{R}^d, \beta_0 \in \mathbb{R}^D, \beta_j \in \mathbb{R}^D$. See Figure 1.

- $\theta = \{[\beta, a_j, b_j, w_j] : j = 1, \ldots, m\}$ denotes the set of model parameters.
- $x_1, \ldots, x_d$ together is called an input layer.
- $A_j := \sigma_j(x) = \sigma(b_j + w_j^\top x)$ is called an activation.
- $A_1, \ldots, A_m$ together is called a hidden layer or hidden unit; m is the number of hidden nodes.
- $f(x)$ is called an output layer.
- g is an output function. Examples are:
  - softmax $g_i(x) = \exp(x_i)/\sum_{l=1}^{D}\exp(x_l)$ for classification. The softmax function estimates the conditional probability $g_i(x) = P(y = i|x)$.
  - identity/linear $g(x) = x$ for regression.
  - threshold $g_i(x) = I(x_i > 0)$
- σ is called an activation function. Examples are:
  - sigmoid $\sigma(x) = 1/(1 + e^{-x})$ (see Figure 2)
  - rectified linear (ReLU) $\sigma(x) = \max\{0, x\}$ (see Figure 2)
  - identity/linear $\sigma(x) = x$
  - threshold $\sigma(x) = I(x > 0)$, threshold gives a direct multi-layer extension of the perceptron (as considered by Rosenblatt).

Activation functions in hidden layers are typically nonlinear, otherwise the model collapses to a linear model. So the activations are like derived features - nonlinear transformations of linear combinations of the features.

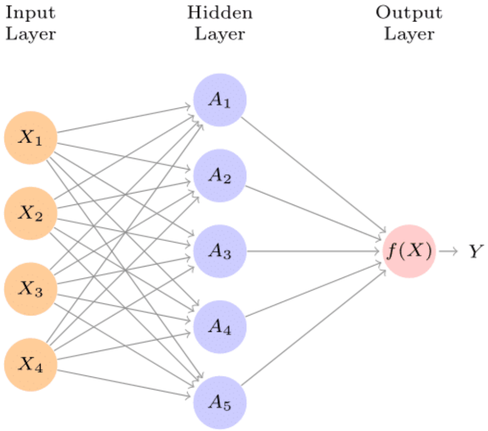

Figure 1: Neural network with a single hidden layer. The hidden layer computes activations $A_j = \sigma_j(x)$ that are nonlinear transformations of linear combinations of the inputs $x_1, \dots, x_d$. Hence these A_j are not directly observed. The functions σ_j are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations A_j as inputs, resulting in a function $f(x)$. Figure 10.1 from [1].

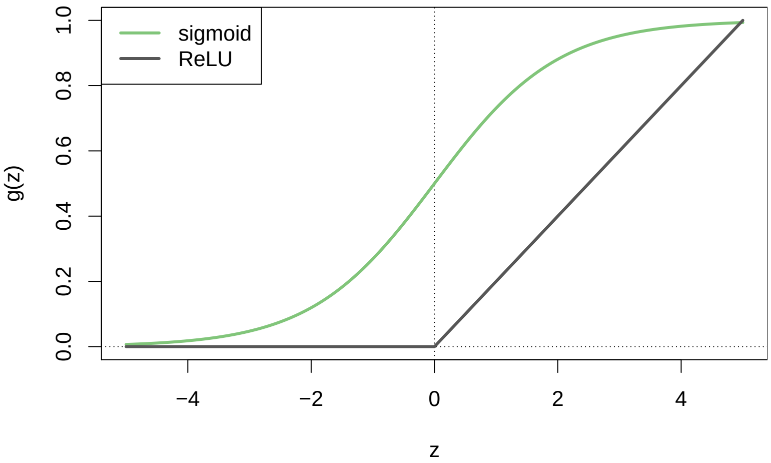

Figure 2: Activation functions. The piecewise-linear ReLU function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison. Figure 10.2 from [1].

1.5 Multi Layer Neural Networks

Modern neural networks typically have more than one hidden layer, and often many units per layer. In theory a single hidden layer with a large number of units has the ability to approximate most functions. However, the learning task of discovering a good solution is made much easier with multiple layers each of modest size.

A deep neural network refers to the model allowing to have more than 1 hidden layers: given input $x \in \mathbb{R}^d$ and response $y \in \mathbb{R}^D$, to predict the response y. K-layer fully connected deep neural network is to build a nonlinear function $f(x)$ as

- Let $m^{(0)} = d$ and $m^{(K)} = D$
- Define recursively

$$\begin{aligned}
&x^{(0)} = x, \quad (x \in \mathbb{R}^{m^{(0)}}), \\
&x_j^{(k)} = \sigma(b_j^{(k)} + (w_j^{(k)})^\top x^{(k-1)}), \quad w_j^{(k)}, x^{(k-1)} \in \mathbb{R}^{m^{(k-1)}}, b_j \in \mathbb{R}^{m^{(k)}}, \qquad k = 1, \ldots, K. \\
&f(x) = g(x^{(K)}).
\end{aligned}$$

- $\theta = \{[b_j^{(k)}, w_j^{(k)}] : k = 1, \ldots, K,\, j = 1, \ldots, m^{(k)}\}$ denotes the set of model parameters.
- $m^{(k)}$ is the number of hidden units at layer k.

2 Notation and Goal

For $k = 1, \ldots, K$, write $b^{(k)} = (b_1^{(k)}, \ldots, b_{m^{(k)}}^{(k)}) \in \mathbb{R}^{m^{(k)}}$, and write $W_k \in \mathbb{R}^{m^{(k)} \times m^{(k-1)}}$ as i-th row of W_k is $(w_j^{(k)})^\top$, i.e.,

$$W_k = \begin{pmatrix} (w_1^{(k)})^\top \\ \vdots \\ (w_{m^{(k)}}^{(k)})^\top \end{pmatrix} \in \mathbb{R}^{m^{(k)} \times m^{(k-1)}},$$

And for $k = 1, \ldots, K-1$, write $\sigma_k : \mathbb{R}^{m^{(k)}} \to \mathbb{R}^{m^{(k)}}$ be coordinatewise application of σ, i.e., $\sigma_k(x_1, \ldots, x_{m^{(k)}}) = (\sigma(x_1), \ldots, \sigma(x_{m^{(k)}}))$, and let $\sigma_K \coloneqq g$. Now, assume $g = \mathrm{id}$. Then K-layer neural network can be described as

$$f_\theta(x) = \sigma_K(W_K \sigma_{K-1}(W_{K-1} \cdots \sigma_1(W_1 x + b^{(1)}) \cdots + b^{(K-1)}) + b^{(K)}).$$

Or inductively,

$$f_\theta^{(0)}(x) = x, \qquad f_\theta^{(k)}(x) = \sigma_k(W_k f_\theta^{(k-1)}(x) + b^{(k-1)}), \qquad f_\theta(x) = f_\theta^K(x).$$

Hence, for K-layer neural network, the function space we consider is $\mathcal{F}_\sigma^{(K)}$, with $\mathcal{F}_\sigma^{(0)} = \{\mathrm{id}\}$ and

$$\mathcal{F}_\sigma^{(k)} = \left\{ f_\theta^{(k)} : f_\theta^{(k)}(x) = \sigma_k(W_k f_\theta^{(k-1)}(x) + b^{(k-1)}),\; f_\theta^{(k-1)} \in \mathcal{F}_\sigma^{(k-1)} \right\}.$$

Suppose the true regression function f_* is in a function class $\mathcal{M}$, so

$$y \approx f_*(x), \qquad f_* \in \mathcal{M}.$$

Suppose are using the ℓ_2-loss, so we find f among deep neural network class $\mathcal{F}$ that minimizes the expected risk (평균위험),

$$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P}\left[(y - f(x))^2\right].$$

f_0 is the expected risk mimizing function (평균위험최소함수). And we estimate f^0 by $\hat{f}$ using data by minimizes on the empirical risk (경험위험) on training dataset, so

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2.$$

$\hat{f}$ is the empirical risk mimizing function (경험위험최소함수). And we set $\tilde{f}$ be the approximation of $\hat{f}$ by optimization(최적화); $\tilde{f}$ is the learned function (학습된 함수).

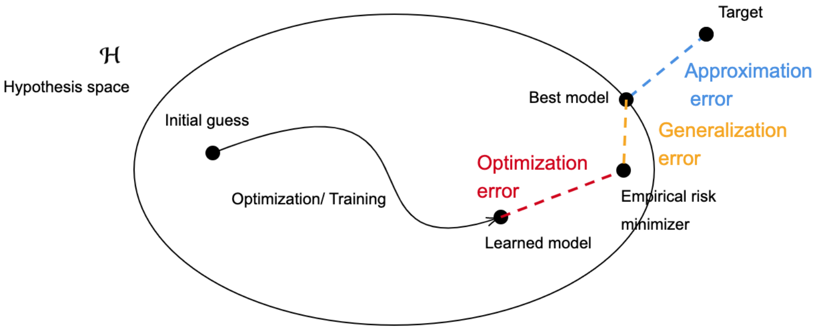

Figure 3: Diagram representing the learning procedure, the three main paradigms and their corresponding errors. Figure 2 from https://dcn.nat.fau.eu/breaking-the-curse-of-dimensionality-with-barron-spaces/.

So there are three sources of errors: approximation error, generalization error, and optimization error. See Figure 3.

$$f_* - \tilde{f} = \underbrace{f_* - f^0}_{\text{approximation error}} + \underbrace{f^0 - \hat{f}}_{\text{generalization error}} + \underbrace{\hat{f} - \tilde{f}}_{\text{optimization error}} .$$

3 Intro: Going Deep

- We have seen that a single hidden layer MLP is sufficient to uniformly approximate continuous functions on a compact set, and can mitigate the curse of dimensinality for some class of smooth functions.
- What if we use more than one hidden layer? What is the role of the depth in this case?

4 Separating shallow and deep networks

Theorem 1 ([2])**.** *For any $K \geq 2$. $f = \Delta^{K^2+2}$ is a ReLU network with $3K^2 + 6$ nodes and $2K^2 + 4$ layers, but any ReLU network g with $\leq 2^K$ nodes and $\leq K$ layers cannot approximate it:*

$$\int_0^1 |f(x) - g(x)|\, dx \geq \frac{1}{32}.$$

Remark 2. (why L_1 metric?)

Previously, we used L_2 and L_∞ to state good upper bounds on approximation; for bad approximation, we want to argue there is a large region where we fail, not just a few points, and that's why we use an L_1 norm.

To be able to argue that such a large region exists, we don't just need the hard function $f = \Delta^{K^2+2}$ to have many regions, we need them to be regularly spaced, and not bunch up. In particular, if we replaced Δ with the similar function $4x(1-x)$, then this proof would need to replace $\frac{1}{32}$ with something decreasing with K.

Proof plans for Theorem 1:

1. (Shallow networks have low complexity.) First we will upper bound the number of oscillations in ReLU networks. The key part of the story is that oscillations will grow polynomially in width, but exponentially in depth.
2. (There exists a regular, high complexity deep network.) Then we will show there exists a function, realized by a slightly deeper network, which has many oscillations, which are moreover regularly spaced. The need for regular spacing will be clear at the end of the proof. We have already handled this part of the proof: the hard function is Δ^{K^2+2}.
3. Lastly, we will use a region-counting argument to combine the preceding two facts to prove the theorem. This step would be easy for the L_∞ norm, and takes a bit more effort for the L_1 norm.

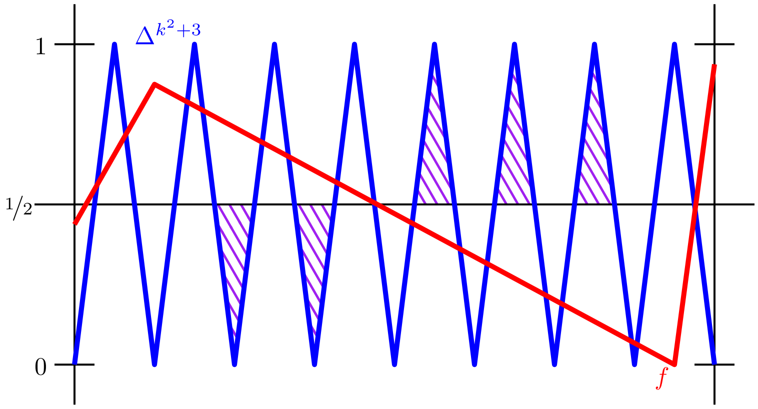

Figure 4: Figure from https://mjt.cs.illinois.edu/dlt/.

Proceeding with the proof, first we want to argue that shallow networks have low complexity. Our notion of complexity is simply the number of affine pieces.

Definition 3. For any univariate function $f : \mathbb{R} \to \mathbb{R}$, let $N_A(f)$ denote the number of affine pieces of f : the minimum cardinality (or ∞) of a partition of $\mathbb{R}$ so that f is affine when restricted to each piece.

Lemma 4. *Let $f : \mathbb{R} \to \mathbb{R}$ be a ReLU network with L layers of widths $(m_1, \ldots, m_K)$ with $m = \sum_i m_i$. Let $g : \mathbb{R} \to \mathbb{R}$ denote the output of some node in layer i as a function of the input. Then the number of affine pieces $N_A(g)$ satisfies*

$$N_A(g) \le 2^i \prod_{j<i} m_j.$$

Hence

$$N_A(f) \le \left(\frac{2m}{K}\right)^K.$$

Remark 5. This immediately hints a "power of composition": we increase the "complexity" multiplicatively rather than additively.

Remark 6. It is natural and important to wonder if this exponential increase is realized in practice. Preliminary work reveals that, at least near initialization, the effective number of pieces is much smaller (Hanin and Rolnick 2019).

This completes part 1 of our proof plan, upper bounding the number of affine pieces polynomially in width and exponentially in depth.

The second part of the proof was to argue that Δ^K gives a high complexity, regular function. For this, consider the Δ function:

$$\Delta(x) = 2\text{Relu}(x) - 4\text{Relu}(x - 1/2) + 2\text{Relu}(x - 1) = \begin{cases} 2x, & 0 \le x \le 1/2, \\ 2 - 2x, & 1/2 \le x < 1, \\ 0 & \text{otherwise.} \end{cases}$$

When $\Delta^K = \Delta \circ \cdots \circ \Delta$ is considered, Δ^K has 2^{K-1} copies of itself, uniformly shrunk down. In a sense, complexity has increased exponentially as a function of the the number of nodes and layers (both $O(K)$). Later, it will matter that we not only have many copies, but that they are identical (giving uniform spacing).

The third part is a counting argument which ensures the preceding two imply the claimed separation in L_1 distance.

The proof proceeds by "counting triangles."

- Draw the line $x \mapsto 1/2$ (as in the figure). The "triangles" are formed by seeing how this line intersects $f = \Delta^{K^2+2}$. There are 2^{K^2+1} copies of Δ, which means $2^{K^2+2} - 1$ (half-)triangles since we get two (half-)triangles for each Δ but one is lost on the boundary of $[0, 1]$. Each (half-)triangle has area $\frac{1}{4} \cdot \frac{1}{2^{K^2+2}} = 2^{-K^2-4}$.

- We will keep track of when g passes above and below this line; when it is above, we will count the triangles below; when it is above, we'll count the triangles below. Summing the area of these triangles forms a lower bound on $\int_{[0,1]} |f - g|$
- Using the earlier lemma, g has $N_A(g) \leq (2 \cdot 2^K / K)^K \leq 2^{K^2}$.
- Let $\mathcal{I}_f \coloneqq \left\{x \in [0,1] : f(x) = \frac{1}{2}\right\}$ and $\mathcal{I}_g \coloneqq \left\{x \in [0,1] : g(x) = \frac{1}{2}\right\}$. Note that

  $$\mathcal{I}_f = \left\{(1) \cdot 2^{-K^2-3}, (3) \cdot 2^{-K^2-3}, \ldots, (2^{K^2+3} - 1) \cdot 2^{-K^2-3}\right\}$$

  has 2^{K^2+2} points. Enumerate $\mathcal{I}_g = \{t_1, \ldots, t_N\}$ with $t_1 < \cdots < t_N$, then $N \leq N_A(g)$. Now, define $\{(l_j, u_j)\}_{j=0}^N$ as

  $$\begin{aligned} &l_0 = \min \mathcal{I}_f, \quad l_j = \min\left\{t \in \mathcal{I}_g : t \geq t_j\right\}, \text{ for } j = 1, \ldots, N. \\ &u_j = \max\left\{t \in \mathcal{I}_g : t \leq t_{j+1}\right\}, \text{ for } j = 0, \ldots, N-1, \quad u_N = \max \mathcal{I}_f. \end{aligned}$$

  Then $l_0 \leq u_0 \leq l_1 \leq u_1 \leq \cdots \leq l_N \leq u_N$ holds, and u_{j-1} and l_j can differ at most by 2^{-K^2-2}.
- For $j = 0, \ldots, N$, between l_j and u_j, $(l_j, u_j) \cap \mathcal{I}_g = \emptyset$, and hence g is constantly above $\frac{1}{2}$ or below $\frac{1}{2}$. When g is above $\frac{1}{2}$, (the number of small triangles below $\frac{1}{2}$) is lower bounded by $2^{K^2+2} \frac{u_j - l_j - 2^{-K^2-2}}{2}$. When g is below $\frac{1}{2}$, (the number of small triangles above $\frac{1}{2}$) is also lower bounded by $2^{K^2+2} \frac{u_j - l_j - 2^{-K^2-2}}{2}$. Hence

  $$\begin{aligned} \int_{l_j}^{u_j} |f - g| &\geq [\text{number of surviving triangles}] \cdot [\text{area of triangle}] \\ &\geq 2^{K^2+1}(u_j - l_j - 2^{-K^2-2}) \cdot 2^{-K^2-4} = 2^{-3}(u_j - l_j - 2^{-K^2-2}). \end{aligned}$$

  Hence together

  $$\begin{aligned} \int_0^0 |f - g| &\geq \sum_{j=0}^N \int_{l_j}^{u_j} |f - g| \\ &\geq 2^{-3} \sum_{j=0}^N (u_j - l_j - 2^{-K^2-2}) \\ &= 2^{-3} \left(u_N - l_0 - \sum_{j=1}^N (u_j - l_j) - N 2^{-K^2-2}\right) \\ &= 2^{-3} \left(\left(1 - 2^{-K^2-2}\right) - N 2^{-K^2-2} - N 2^{-K^2-2}\right) \\ &= 2^{-3} \left(1 - (2N+1) 2^{-K^2-2}\right). \end{aligned}$$

- Now use the relation $N \leq N_A(g) \leq 2^{K^2}$ to conclude

  $$\begin{aligned} \int_0^0 |f - g| &\geq 2^{-3} \left(1 - (2^{K^2+1} + 1) 2^{-K^2-2}\right) \\ &\geq 2^{-3} \left(1 - \frac{1}{2} - 2^{-K^2-2}\right) \\ &\geq \frac{1}{32}. \end{aligned}$$

5 Uniform approximation by deep networks

Consider the case approximating $f : [0,1]^d \to \mathbb{R}$ by a ReLU MLP (Multi-Layer Perceptron). If $\hat{f}$ approximates f, we let

$$\|\hat{f} - f\|_\infty = \sup_{x \in [0,1]^d} |\hat{f}(x) - f(x)|.$$

We consider K-layer MLP. A single hidden layer MLP corresponds to $K = 2$.

Theorem 7 ([4])**.** *Suppose $d, \beta \in \mathbb{N}$ and $\epsilon \in (0,1)$. (In fact, $\beta \geq 1$ is enough) Suppose $f \in H_d(\beta, 1)$. Then there is an K-layer ReLU MLP $\hat{f}$ such that*

$$\left\| \hat{f} - f \right\|_\infty < \epsilon,$$

with $K \leq c(\log(1/\epsilon)+1)$ and at most $c\epsilon^{-d/\beta}(\log(1/\epsilon)+1)$ number of weights and activation units, where $c = c(d, \beta)$.

Theorem 8 ([4])**.** *Let $f \in \mathcal{C}^2([0,1]^d)$ be a nonlinear function (i.e., not of the form $f(x_1, \ldots, x_d) \equiv a_0 + \sum_{k=1}^d a_k x_k$ on the whole $[0,1]^d$). Then, for any fixed K, a depth-K ReLU MLP approximating f with error $\epsilon \in (0,1)$ must have at least $c\epsilon^{-1/[2(K-1)]}$ weights and activation units, with some constant $c = c(f, K) > 0$.*

- Functions in the Hölder space $H_d(\beta, 1)$ can be ϵ-approximated by a ReLU MLP with depth $O(\log(1/\epsilon))$ and the number of activation units $O(\epsilon^{-d/\beta} \log(1/\epsilon))$.
- In contrast, a nonlinear function from $\mathcal{C}^2([0,1]^d)$ *cannot* be ϵ-approximated by a ReLU MLP of *fixed* depth K with the number of units less than $\Omega(\epsilon^{-1/[2(K-1)]})$.
- $H_d(\beta, 1) \subset \mathcal{C}^2([0,1]^d)$ if $\beta > 2$. In this case, in terms of the required number of activation units, unbounded-depth approximation of functions in $H_d(\beta, 1)$ is asymptotically strictly more efficient than approximations with a fixed depth L when
$$\frac{d}{\beta} < \frac{1}{2(K-1)}.$$
which is true if $\beta \gg d$, i.e., the target function is smooth enough.
- The efficiency of depth is even more pronounced for very smooth functions such as polynomials, which can be implemented by deep networks using only $O(\log(1/\epsilon))$ units.

6 Approximating x^2

We in particular look at x^2. We will have something like following:

Theorem 9 ([4])**.** *The function $f(x) = x^2$ on $[0,1]$ can be approximated with any error $\epsilon > 0$ by a ReLU network having the depth and the number of weights and computation units $O(\log(1/\epsilon))$.*

Why x^2?

- Why it should be easy: because $x^2 = \int_0^\infty 2\sigma(x-b)db$, so we need only to uniformly place ReLUs.
  - We'll use an approximate construction due to (Yarotsky 2016). It will need only poly log poly $\log(1/\epsilon)$ nodes and depth to ϵ-close.
  - By contrast, our shallow univariate approximation theorems needed $1/\epsilon$ nodes.
- Why we care: with x^2, polarization gives us multiplication:
$$xy = \frac{1}{2}\left((x+y)^2 - x^2 - y^2\right).$$
From that, we get monomials, polynomials, Taylor expansions.

Define $S_i := \left(\frac{0}{2^i}, \frac{1}{2^i}, \ldots, \frac{2^i}{2^i}\right)$; let h_i be the linear interpolation of x^2 on S_i. See Figure 5.

Then:

- $h_i = h_{i+1}$ on S_i.
- For $x \in S_{i+1} \backslash S_i$, defining $\epsilon = 2^{-i-1}$,
$$\begin{aligned} h_i(x) - h_{i+1}(x) &= \frac{1}{2}\left(h_i(x-\epsilon) + h_i(x+\epsilon)\right) - h_{i+1}(x)) \\ &= \frac{1}{2}\left((x-\epsilon)^2 + (x+\epsilon)^2\right) - x^2 = \epsilon^2. \end{aligned}$$
Key point: no dependence on x.

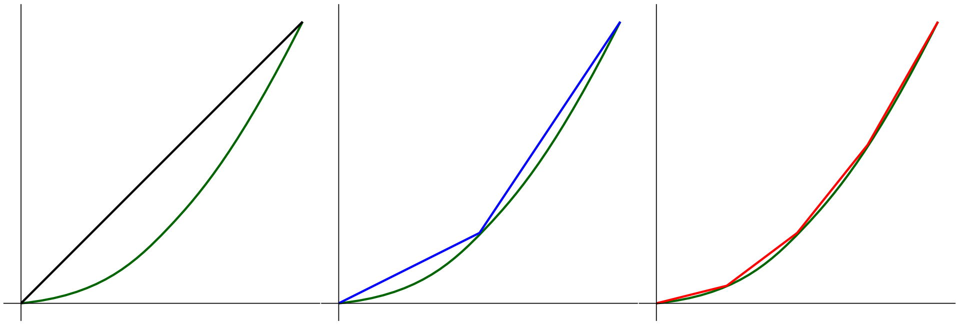

Figure 5: Figure from https://mjt.cs.illinois.edu/dlt/.

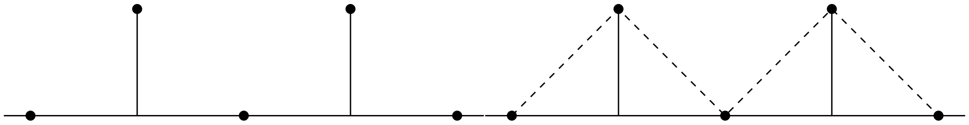

Figure 6: Figure from https://mjt.cs.illinois.edu/dlt/.

- For any $x \in S_{i+1}$,
$$h_{i+1}(x) = h_i(x) - \frac{1}{4^{i+1}} 1(x \in S_{i+1} \backslash S_i).$$

- Since h_{i+1} linearly interpolates, then $h_{i+1} - h_i$ must also linearly interpolate. See Figure 6. The linear interpolation of $1[x \in S_{i+1} \backslash S_i]$ is Δ^{i+1}. Thus
$$h_{i+1} = h_i - \frac{\Delta^{i+1}}{4^{i+1}}.$$

- Since $h_0(x) = x$, then $h_i(x) = x - \sum_{j=1}^{i} \frac{\Delta^j(x)}{4^j}$.

Theorem 10 ([4]). *1. h_i is the piecewise-affine interpolation of x^2 along $[0, 1]$ with interpolation points S_i.*

2. h_i can be written as a ReLU network consisting of $2i$ layers and $3i$ nodes using "skip connections," or a pure ReLU network with $2i$ layers and $4i$ nodes.

3. $\sup_{x \in [0,1]} \left| h_i(x) - x^2 \right| \leq 4^{-i-1}$.

4. Any ReLU network f with $\leq K$ layers and N nodes satisfies
$$\int_{[0,1]} (f(x) - x^2)^2 dx \geq \frac{1}{5760(2N/K)^{4K}}.$$

From squaring we can get many other things (still with $O(\log(1/\epsilon))$ depth and size.

- Multiplication (via "polarization"):
$$(x, y) \mapsto xy = \frac{1}{2}\left((x + y)^2 - x^2 - y^2\right).$$

- Multiplications gives polynomials.

- $\frac{1}{x}$ and rational functions ([3])

- Functions with "nice Taylor expansions" (Sobolev spaces) (Yarotsky 2016); though now we'll need size bigger than $\log\left(\frac{1}{\epsilon}\right)$.

References

[1] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning—with applications in R.* Springer Texts in Statistics. Springer, New York, [2021] ©2021. Second edition [of 3100153].

[2] Matus Telgarsky. Benefits of depth in neural networks. *CoRR*, abs/1602.04485, 2016.

[3] Matus Telgarsky. Neural networks and rational functions. *CoRR*, abs/1706.03301, 2017.

[4] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 10 2017.