ML Theory Lecture 7

Matus Telgarsky

1 Depth hierarchy theorem for neural nets

Recall that a function f is piecewise affine when there exists a partition of $\mathbb{R}$ into intervals so that f is affine within each pieces; let $N_A(f)$ denote the minimum number of pieces in this partition (possibly $N_A(f) = \infty$), and let $P_A(f)$ be some partition with $N_A(f) = |P_A(f)|$ (note that $P_A(f)$ is not unique).

We concluded last lecture with a 4 part lemma, the key part of which was an upper bound on the number of affine pieces in a single neural network node.

Lemma 1.1. *Let univariate functions $f, g, (g_1, \ldots, g_t)$ and scalars $(a_1, \ldots, a_t, b)$ be given. Then*

$$N_A\left(x \mapsto f(\sum_i a_i g_i(x) + b)\right) \le N_A(f) \cdot \sum_i N_A(g_i).$$

Invoking this lemma inductively gives a bound on $N_A(f)$ where $f : \mathbb{R} \to \mathbb{R}$ is a neural net.

Theorem 1.2. *Let $f : \mathbb{R} \to \mathbb{R}$ be the function computed by a neural network with L layers, every activation σ satisfies $N_A(\sigma) \le t$, and layer i has N_i nodes, with $N := \sum_i N_i$ for convenience. The following bounds hold.*

1. *Consider any node in layer i, and let $g : \mathbb{R} \to \mathbb{R}$ denote the computation of this node as a function of the network input. Then $N_A(g) \le t^i \prod_{j<i} N_j$.*

2. $N_A(f) \le \left(\frac{tN}{L}\right)^L$.

Remark 1.3.
- We will establish this bound via elementary means; in the more general case of multivariate inputs, VC arguments can be adapted to give similar bounds.
- As a sanity check, let's consider $N_A(\Delta^k)$. We know that Δ^k is 2^{k-1} copies of Δ, meaning along $[0, 1]$ it consists of 2^k distinct affine functions. Account for the behavior outside this interval,

  $$N_A(\Delta^k) = 2 + 2^k.$$

  Let's also prove it via the preceding theorem. The construction uses $2k$ layers and $3k$ nodes, and moreover $N_A(\sigma_r) = 2$, thus

  $$N_A(f) \le \left(\frac{2 \cdot 3k}{2k}\right)^{2k} = 9^k.$$

  Upon further inspection, the Δ^k construction can remove the layers with single nodes and make use of $k + 1$ layers, giving the tighter estimate 6^{k+1}.

  We are also losing some factors because we didn't require piecewise affine functions to be continuous.

  Taking all this together, Δ^k is pretty efficient at meeting the bound. This is essential because we want N_A to be a measure of complexity of neural networks which is small for shallow networks and not only large but also roughly tight for Δ^k.

◇

Proof. First note that the second claim follows from the first. Indeed, the output node, as a function of the input, computes f, thus $N_L = 1$ implies

$$N_A(f) \le t^L \prod_{j \le i} N_j.$$

The bound follows by considering the worse case for $\prod_{j<i} N_j$; this can be bounded in various ways, one being Jensen's inequality:

$$\prod_{j \le L} N_j = \exp \sum_{j \le L} \ln N_j = \exp \frac{1}{L} \sum_{j \le L} L \ln N_j \le \exp L \ln \sum_{j \le L} \frac{N_j}{L} = \left(\frac{N}{L}\right)^L.$$

This bound is almost attained by making all nodes by making all layers have the same number of nodes (and this solution can be grinded out via the Lagrangian); it's only "almost" because $N_L = 1$.

Let's turn to proving the first part via induction on layers. The induction will use the simplifying trick of starting from layer 0, the first input; for this reason, define $N_0 := 1$, which does not change the product term $\prod_{j<i} N_j$.

For that base case, there is nothing to show; the input is an affine function of the input (identity mapping), thus the number of pieces is $1 = t^0 \prod_{j<0} N_j$.

For the inductive step, suppose the nodes in layer i, treated as functions of the network input, compute $(g_1, \ldots, g_{N_i})$ with

$$N_A(g_j) \le t^i \prod_{j<i} N_j.$$

Now consider any node in layer $i+1$, and let g denote its output as a function of the network, and let σ denote its activation. Combining the preceding inductive hypothesis with Lemma 1.1,

$$N_A(g) \le N_A(\sigma) \sum_{j=1}^{N_i} N_A(g_j) \le t \sum_{j=1}^{N_i} t^i \prod_{j<i} N_j \le t^{i+1} \prod_{j<i+1} N_j.$$

□

Combining this estimate with the structure of Δ^k from the last lecture gives the following separation result (called a "depth hierarchy theorem" in TCS).

Theorem 1.4 (Telgarsky (2015, 2016))**.** *Let any integer $k \ge 2$ be given. Then the function Δ^{k^2+3} can be represented as a ReLU network with $3k^2+9$ total nodes and $2k^2+6$, however any function f represented as a ReLU network with $\le 2^k$ nodes and $\le k$ layers can not approximate it in L_1:*

$$\left\|\Delta^{k^2+3} - f\right\|_1 = \int_{[0,1]} \left|\Delta^k(x) - f(x)\right| \mathrm{d}x \ge \frac{1}{32}.$$

Remark 1.5.
- Note that result has various inefficiences: we want to compare k-layered functions to $(k+1)$-layered functions rather than $(2k^2+6)$-layered functions; $1/32$ should be $1/2 - o(1)$; we only exhibited one hard function, rather than many, or discussing natural functions (for instance as found by sgd); the bound has only combinatorial quantities and no sensitivity to weight magnitudes.
- The proof will use N_A to essentially count oscillations, however just as in Homework 0, this will not suffice: we will need the *regularity* of Δ^{k^2+3}'s oscillations.
- We preferred $\|\cdot\|_\mathrm{u}$ for upper bounds, but for lower bounds $\|\cdot\|_1$ is better; it tells us that we can't get close to the target function for a decent fraction of the space.

- It is essential that the right hand side is a constant, independent of k.

$\diamond$

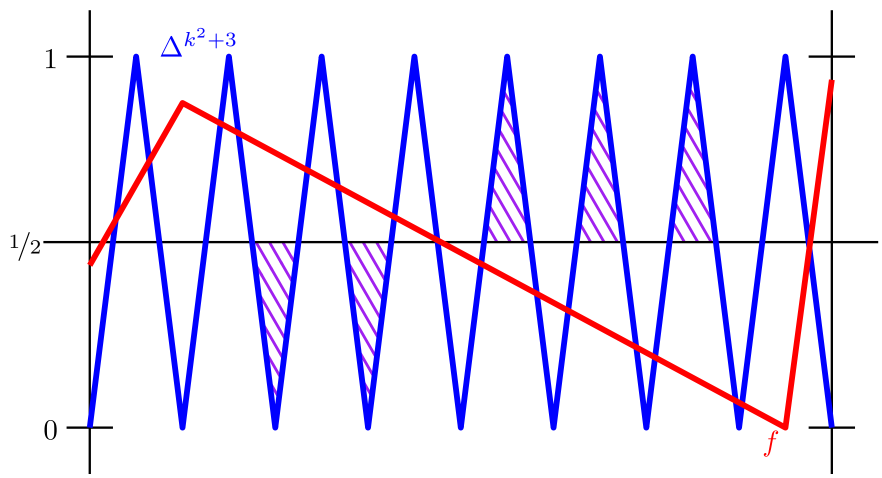
Proof. The lemma in the last lecture established that Δ^{k^2+3} consists of k^2+2 copies of Δ, uniformly squeezed to fit within $[0,1]$; this written compactly as

$$\Delta^{k^2+3}(x) = \Delta\left(\left\langle 2^{k^2+2}x\right\rangle\right).$$

On the other hand, suppose f has $\le 2^k$ nodes and $L \le k$ layers; Theorem 1.2 tells us

$$N_A(f) \le \left(\frac{2\cdot 2^k}{L}\right)^L \le \left(\frac{2\cdot 2^k}{k}\right)^k \le 2^{k^2},$$

where the substitution of L with k is due to $L=k$ maximizing the expression, for instance as can be determined by differentiating. Let's put everything we just established into a single plot of Δ^{k^2+3} and f.



This plot has some parts shaded in. Recall that our goal is to lower bound the L_1 distance between f and Δ^{k^2+3}. Inspecting the plot, a lower bound can be constructed as follows:

- Subdivide $[0,1]$ into regions according to f being either above or below $1/2$.
- Let's split Δ^{k^2+3} by $x \mapsto 1/2$, obtaining $2^{k^2+3}-1$ triangles (we lose one at the boundaries).
- Whenever f is above $1/2$, we can count the triangles below $1/2$; analogously, when f is below $1/2$, count the triangles above $1/2$.
- By construction, the total area in these triangles lower bounds the L_1 distance.

In order to count these triangles, let's be a little careful to avoid double counting. Let's use the following scheme to ignore certain triangles, which will give a valid lower bound and also corresponds to the above shading.

- First, cross off all triangles at the boundary of a piece of f, meaning an interval in $P_A(f)$. Consequently, $N_A(f)$ triangles are removed. (Note: we need to do this because we didn't require f to be continuous; the boundary of a piece can thus trigger a jump across $1/2$.)
- Within each interval of $P_A(f)$, f is affine, thus additionally cross off any triangle where f crosses $1/2$, meaning $N_A(f)$ additional triangles are removed.

- At this point, $2 \cdot N_A(f)$ triangles are crossed off. Consider the contiguous groups of uncrossed triangles; if any group has odd cardinality, cross off a single endpoint, thus leaving an even number of triangles. This crosses off at most $2 \cdot N_A(f)$ additional triangles.
- The remaining contiguous pieces of triangles all now denote regions where f is either bounded below by $1/2$, or bounded above by $1/2$. Thus cross off half of all unmarked triangles, those on the same side of $1/2$ as f; the remaining triangles can all be shaded in, and are guaranteed to not cross f.

After all these operations,

$$\#\text{triangles} \geq \frac{1}{2}\left(2^{k^2+3} - 1 - 4N_A(f)\right) \geq 2^{k^2+2} - \frac{1}{2} - 2^{k^2+1} \geq 2^{k^2}.$$

Thus

$$\begin{aligned}\int_{[0,1]} |\Delta^{k^2+3}(x) - f(x)|\,\mathrm{d}x &\geq \left[\#\text{triangles}\right] \cdot \left[\text{triangle area}\right] \\ &\geq \left[\frac{1}{2}\left(2^{k^2+3} - 1 - 4N_A(f)\right)\right] \cdot \left[\frac{1}{4} \cdot \frac{1}{2^{k^2+3}}\right] \\ &\geq \left[2^{k^2+3} - 1 - 4 \cdot 2^{k^2}\right] \cdot \left[\frac{1}{2^{k^2+6}}\right] \\ &\geq \frac{2^{k^2+1}}{2^{k^2+6}} = \frac{1}{32}.\end{aligned}$$

□

Before adjourning this section, let's point out some crucial prior work.

- Håstad (1986) gave the classic depth hierarchy theorem for boolean circuits, using both parity and "Sipser Functions" as hard functions. Similarly to the above result, there was a gap between the depth of the hard function and the shallow functions.
- Rossman et al. (2015) resolved a few issues in Håstad's result, namely: the depth gap between hard and comparison circuits was just 1, and the error lower bound was $1/2 - o(1)$. The construction used the proof technique due to Håstad (1986), and the hard functions were a variant of the Sipser functions.
- Eldan and Shamir (2015) showed that there exist 3-layer neural networks which can not be approximated by 2-layer networks unless they have 2^d times as many nodes. Recently, Daniely (2017) provided a vastly simplified proof.

2 Squaring with neural nets

[We started this topic; we'll do it in detail next lecture.]

References

Amit Daniely. Depth separation for neural networks. In *COLT*, 2017.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. 2015. `arXiv:1512.03965 [cs.LG]`.

Johan Håstad. *Computational Limitations of Small Depth Circuits*. PhD thesis, Massachusetts Institute of Technology, 1986.

Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. In *FOCS*, 2015.

Matus Telgarsky. Representation benefits of deep feedforward networks. 2015. `arXiv:1509.08101v2 [cs.LG]`.

Matus Telgarsky. Benefits of depth in neural networks. In *COLT*, 2016. `arXiv:1602.04485v1 [cs.LG]`.