10-707: Advanced Deep Learning **Spring 2020**

Lecture 3: January 22

Lecturer: Andrej Risteski *Scribes: Shuhao Zhang, Arundhati Banerjee*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

1 Introduction

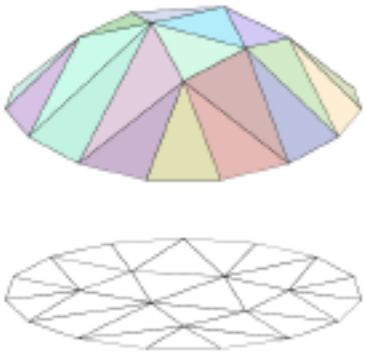
In last class we showed the proof of universal approximation theorem, which states that any reasonably nice function can be approximated by a 3-layer ReLU neural network. However, there is some functions that shallow networks cannot approximate well but deeper networks can do(e.g. a deep network cannot be approximated by a shallow one). Also, it is more possible for people to extract more high-level information from a trained deeper networks than shallow networks.
In this lecture, we will show the limitation of shallow network by first looking at it's graph, then build a function that shallow network cannot approximate.

2 Linear pieces of shallow functions

2.1 The graph of ReLU network

A ReLU network f is piecewise linear, which means we can subdivide domain of f into finite number of polyhedral pieces $(P_1, P_2, \ldots\ldots, P_N)$,s.t. in each piece, f is linear, in other words, $\forall x \in P_i, f(x) = A_i x + b_i$

Here is an example of a ReLU network f with two input x_1, x_2, the cap like graph above the domain plane is the graph of $f(x_1, x_2)$, we can see that by dividing the round-like domain into pieces, all pieces of the graph of value are planes, which means they can be represented as $A_i X + b_i$.



Therefore, if a ReLU network is merely a piecewise linear function with finite pieces, at least we can say that it cannot precisely approximate a piecewise linear function with more lineas piece than itself. So the next step is to see for a specific ReLU network, how many linear pieces it can have.

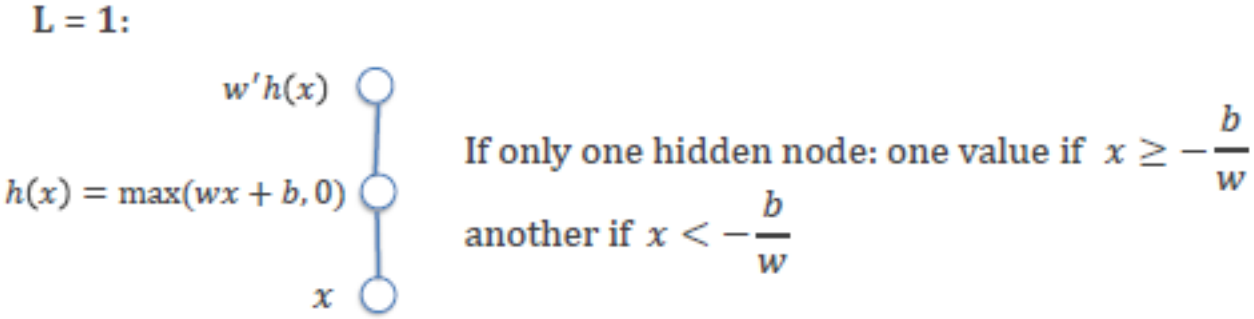
2.2 Number of Linear pieces

Claim: If $f : R \to R$ is a ReLU network with L hidden layers widths $(m_1, m_2, \ldots\ldots m_L)$, then f has at most $2^{L-1}(m_1 + 1)m_2 \ldots\ldots m_L$ linear pieces.

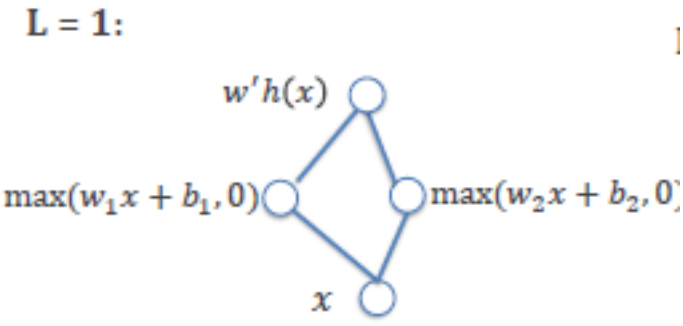
Proof:

Using mathematical induction method, let's see the case that a univariate ReLU network with only 1 hidden layer and 1 node:

$L = 1, m_1 = 1$:

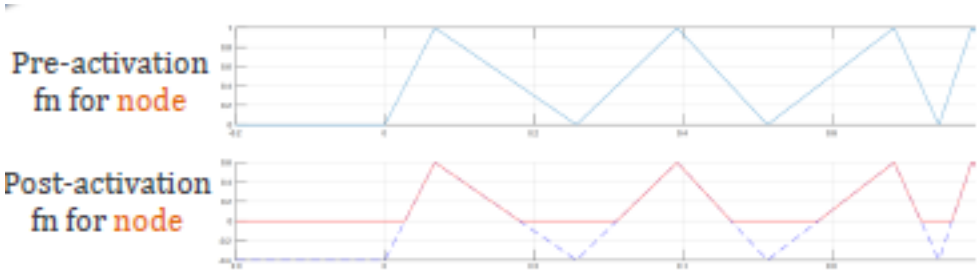


It's easy to see that there are two linear pieces, $w' * 0 + b'$ when $x \in [-\infty, -\frac{b}{w}]$ and $w' * (wX + b) + b'$ when $x \in [-\frac{b}{w}, \infty]$ $L = 1, m_1 = 2$:

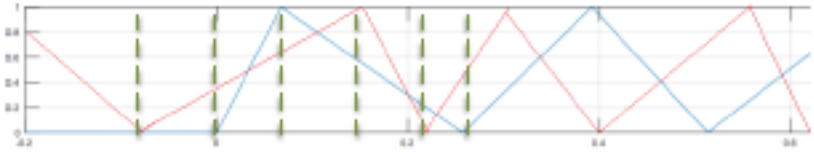


When $m_1 = 2$, there are three linear pieces(assume $-\frac{b_2}{w_2} \geqslant -\frac{b_1}{w_2}$) doamin on $[-\infty, -\frac{b_1}{w_1}], [-\frac{b_1}{w_1}, -\frac{b_2}{w_2}], [-\frac{b_2}{w_2}, \infty]$
It is easy to conclude that the first hidden layer with m_1 nodes can at most seperate the whole domain into $m_1 + 1$ linear pieces.
Then, we add another hidden layer L_2 and see what effect a single node in L_2 can have:

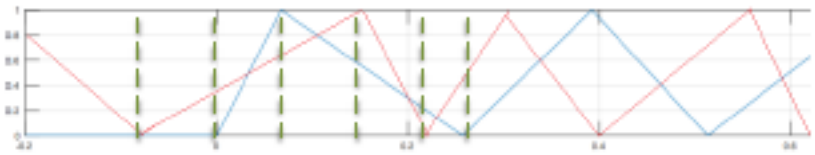
For a specific node in L_2 (say l_{21}), assume it's input from penultimate layer is $f_1(x)$, as we discussed before, $f_1(x)$ have at most $m_1 + 1$ linear pieces, the output of node l_{21}) is $max\{w_{21}f_1(x) + b_{21}, 0\}$, where w_{21} is a row vector from the coefficient matrix W_{12} between layer L_1 and L_2.
$w_{21}f_1(x) + b_{21}$ will have same number of linear pieces as $f_1(x)$ since it's merely add a linear transformation to $f_1(x)$, the major effect is from ReLU function:



From this graph we can see that the only chance for ReLU function to change the number of linear pieces is when there is positive and negative changes in a piece. And if it happened, this linear piece will be truncated into two linear pieces at zero point. Therefore, by implementing ReLU function, the number of linear pieces in the output is at most twice as the input.
Then, when should see what will happen when adding all outputs from nodes in L_2:



The graph above show the process of adding two piece-wise linear functions together, we can conclude that if no function has an inflection point(non-derivable point) in a certain interval, then this interval will be a linear piece after adding two functions together. So for two piece-wise linear functions, if all of their non-derivable point are different there will be a+b linear pieces, and if some of their non-derivable points are same, there will be less linear pieces.



Therefore, suppose the output of penultimate layer L_{k-1} have p_{k-1} linear pieces, after the treatment of layer L_k, the output have at most $2p_{k-1}m_k$, where m_k is the number of nodes in L_k.

In summary, a ReLU network with L hidden layers widths $(m_1, m_2, \ldots\ldots m_L)$ can at most have $2^{L-1}(m_1 + 1)m_2 \ldots\ldots m_L$ linear pieces in it's output.

3 Deep function with many oscillations

First we study the triangle function (Fig. 3.1) Δ defined as:

$$\Delta : [0,1] \to \mathbf{R}, \quad \Delta(x) = 2\sigma(x) - 2\sigma(2x-1) = \begin{cases} 2x, & x \leq \frac{1}{2} \\ 2 - 2x, & x > \frac{1}{2} \end{cases}$$

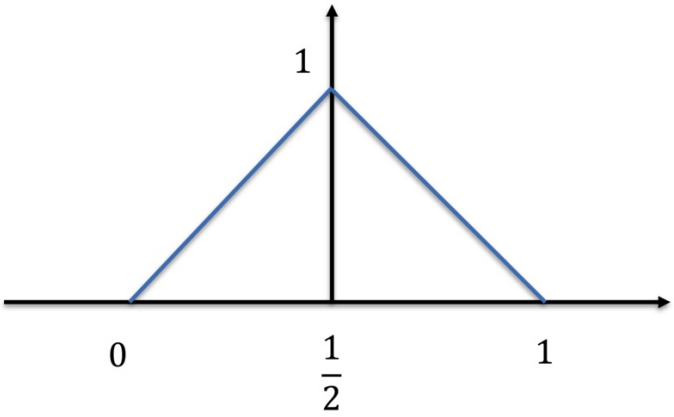


Figure 3.1: Triangle function

Note that σ here is the ReLU activation function $\sigma(x) = max\{0, x\}$.

The interesting property of this function is that composing Δ with itself squishes in copies of itself to look like a sawtooth with 2^{k-1} peaks for k self-compositions. (Fig. 3.2). We now prove this formally by induction on k.

Claim: $\Delta^k(x) = \Delta(2^{k-1}x - \lfloor 2^{k-1}x \rfloor)$

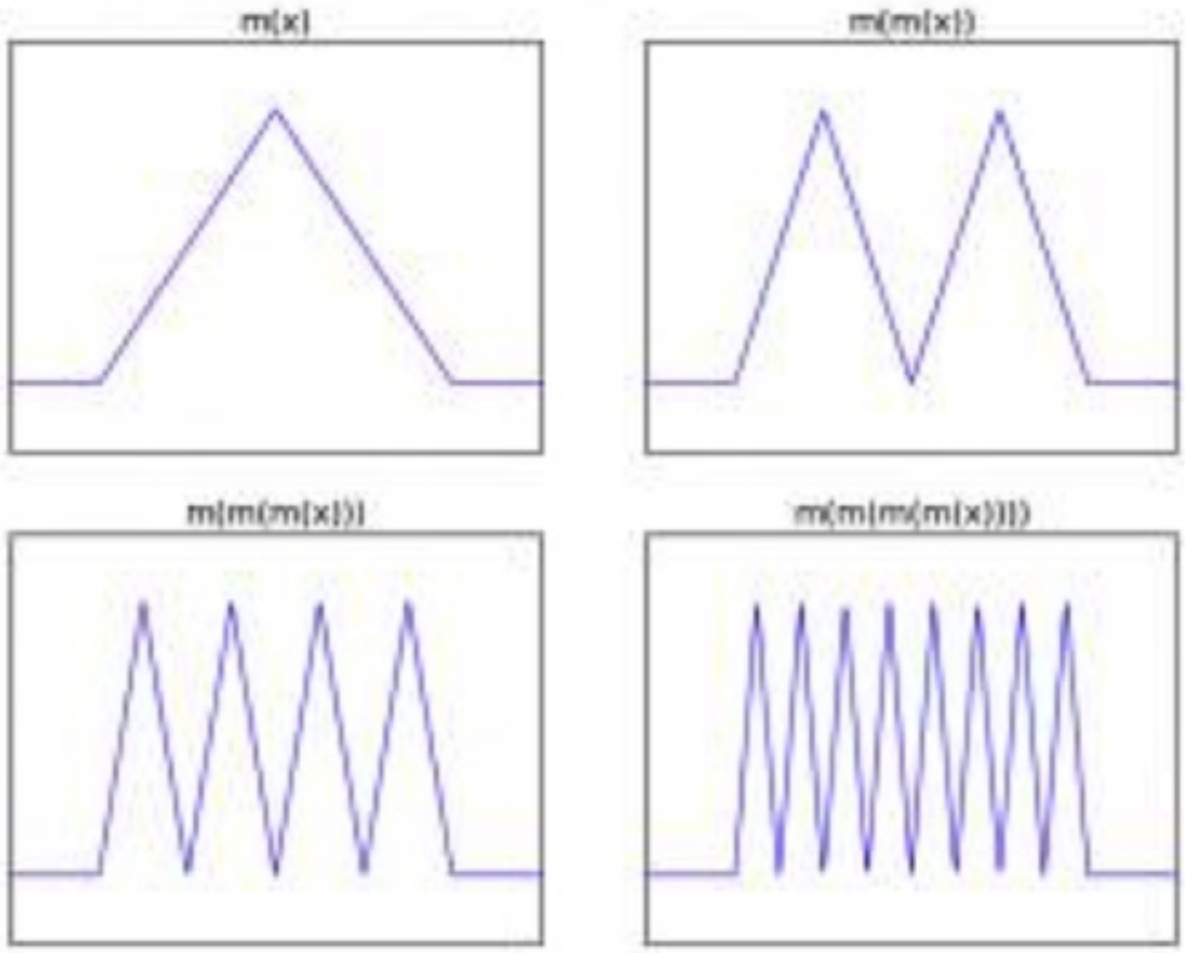


Figure 3.2: Composing Δ with itself

(The expression in braces is the fractional part of $2^{k-1}x$ and the RHS intuitively corresponds to squishing a triangle in every $\frac{1}{2^{k-1}}$ interval in $[0,1]$ since $\Delta(x)$ squishes in 2^0 i.e. one triangle in $[0,1]$)

Proof: (by induction)
Base case: k=1 - follows by definition since $x \in [0,1]$ so $\Delta(x) = \Delta(x - \lfloor x \rfloor)(= \Delta(x-0))$
Inductive step: Assuming it holds for some $k > 1$, we try to prove our claim for $k+1$.
If $x \leq \frac{1}{2}$:

$$\Delta^{k+1}(x) = \Delta^k(\Delta(x)) \tag{3.1}$$
$$= \Delta^k(2x) \quad \text{using definition} \tag{3.2}$$
$$= \Delta(2^{k-1}(2x) - \lfloor 2^{k-1}(2x) \rfloor) \quad \text{by induction} \tag{3.3}$$
$$= \Delta(2^k x - \lfloor 2^k x \rfloor) \tag{3.4}$$

If $x > \frac{1}{2}$:

$$\Delta^{k+1}(x) = \Delta^k(\Delta(x)) \tag{3.5}$$
$$= \Delta^k(2-2x) \quad \text{using definition} \tag{3.6}$$
$$= \Delta^{k-1}(\Delta(2-2x)) \tag{3.7}$$
$$= \Delta^{k-1}(\Delta(1-(2-2x))) \quad \text{since } \Delta \text{ is symmetric about } x = \frac{1}{2} \text{ so } \Delta(x) = \Delta(1-x) \tag{3.8}$$
$$= \Delta^{k-1}(\Delta(2x-1)) \tag{3.9}$$
$$= \Delta^k(2x-1) \tag{3.10}$$
$$= \Delta(2^{k-1}(2x-1) - 2^{k-1}(2x-1)\rfloor) \text{ using induction} \tag{3.11}$$
$$= \Delta(2^k x - 2^{k-1} - \lfloor 2^k x - 2^{k-1} \rfloor) \tag{3.12}$$
$$= \Delta(2^k x - 2^{k-1} - \lfloor 2^k x \rfloor + 2^{k-1}) \text{ since } 2^{k-1} \text{ is an integer} \tag{3.13}$$
$$= \Delta(2^k x - \lfloor 2^k x \rfloor) \tag{3.14}$$

Our goal is to build a function using Δ and show that while deep networks can approximate it, shallow networks lack the ability to do so (unless they pay in size), thereby making a point in favour of the necessity of deep networks. We consider our target function f to be $\Delta^{2L^2+2}(x)$ (where we will relate L with the depth of the network).

Theorem(Telgarsky'15): For every $L \in \mathbb{N}$, there is a function $f : [0,1] \to [0,1]$ representable as a network of depth $O(L^2)$ with $O(L^2)$ nodes and ReLU activation such that for every network $g : [0,1] \to \mathbb{R}$ of depth L and $\leq 2L^2$ nodes, and ReLU activation we have $\int_{[0,1]} |f(x) - g(x)| dx \geq \frac{1}{32}$
Note:

- Since we are proving a lower bound so using the L1 error in the average sense gives a stronger notion of the approximation inability of the shallow network than L∞ which we used in last class.

3.1 Shallow g's can't approximate f

We use the Fig. 3.3 for reference. Our aim is to find the error in approximation. Consider the line $y = \frac{1}{2}$ that divides each triangle of f into 2 half triangles. We want to count the number of half triangles of f on opposite side than g. Each of them will contribute $\frac{1}{2}\times$(triangle area) to the integral, since

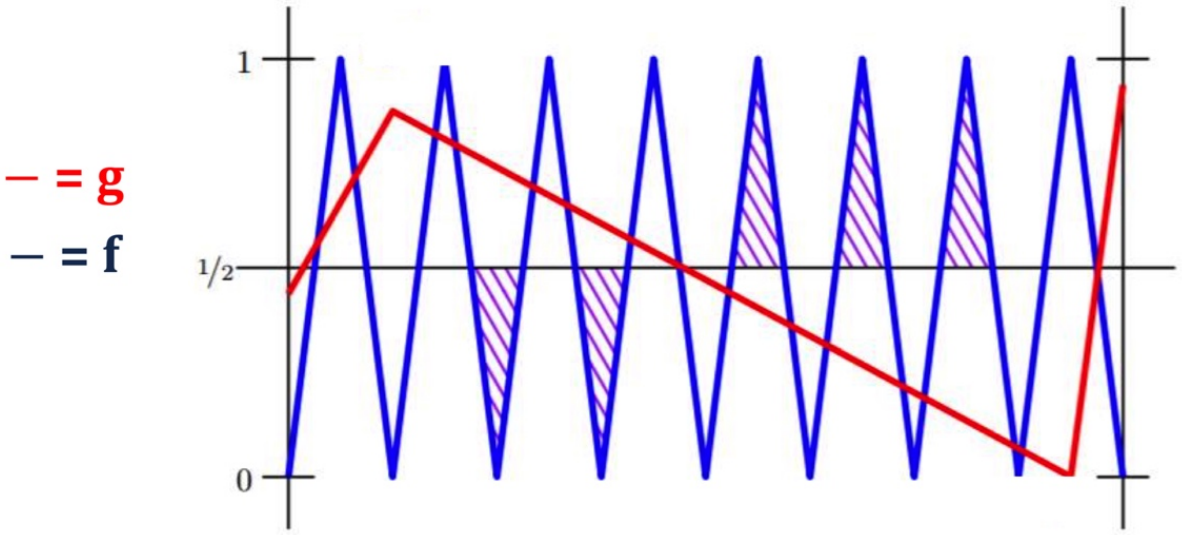


Figure 3.3: Proof by picture

$\int_{x\in triangle} |f-g|dx \geq \int_{x\in triangle} \frac{1}{2}dx = \frac{1}{2}\times$(triangle area). Since our target function is Δ^{2L^2+2}, from our previous study of the behaviour of the delta function on composing with itself, we will have number of half triangles in f given by $N_f = 2\times 2^{2L^2+2-1} - 1 = 2^{2L^2+2} - 1$. We subtract 1 because of one half triangle being lost at the edge. Consider N_g to be the number of half triangles of g.

For any interval spanned by a half triangle from N_g, if there are m triangles of f contained in that interval, then at least $\frac{m-1}{2}$ half triangles of f will be on the opposite side than the half triangle of g (because the half triangles in f alternate symmetrically and $\lceil \frac{m}{2} \rceil \geq \lfloor \frac{m}{2} \rfloor \geq \frac{m-1}{2}$). Since we might lose some triangles of f at the boundary between the half triangles of g, we can lose at most N_g of those. So,

$$\sum_{1}^{N_g} \frac{m-1}{2} = \frac{\sum_{i=1}^{N_g} m_i - N_g}{2} = \frac{N_f - N_g - N_g}{2} = \frac{N_f - 2N_g}{2}$$

where $\sum_i m_i$ is the number of triangles of f contained in g which can be at most $N_f - N_g$ due to the ones lost at the boundary as described before. So we can have at least $\frac{N_f - 2N_g}{2}$ half triangles of f on opposite side than g. Next we need to relate this to our approximation error.

Using the result on the number of linear pieces representable by a L-layered ReLU network we have $N_g \leq (2n)^L$ taking the number of nodes in each layer, $m_l < n$, the total number of nodes. Since g is a shallow network, we use $n < 2^L$ to get $N_g \leq (2n)^L \leq 2^L \times 2^{L^2} \leq 2^{2L^2}$. Plugging in N_f and N_g we get the number of contributing half triangles $\geq \frac{2^{2L^2+2}-1-2\times 2^{2L^2}}{2} \geq \frac{2^{2L^2+1}-1}{2} \geq 2^{2L^2} - \frac{1}{2} \geq 2^{2L^2} - 1$. Using the formula for the area of the triangle, we can evaluate the integral as follows:

$$\int_{x\in triangle} |f-g|dx \geq \frac{1}{2} \times (2^{2L^2} - 1) \times \frac{1}{2} \times (base) \times (height) \tag{3.15}$$

$$= \frac{1}{2} \times (2^{2L^2} - 1) \times \frac{1}{2} \times \frac{1}{2^{2L^2+2}} \times \frac{1}{2} \tag{3.16}$$

$$\geq \frac{1}{32} \tag{3.17}$$

This proves our theorem.

4 Parting thoughts

- The theorem regarding approximation power of the deep networks can be extended to d dimensions and with other activation functions.
- Depth can help in easing the curse of dimensionality studied in last lecture to some extent. (idea behind Theorem due to Yarotsky '16.)
- Different architectures like ResNet, DenseNet have been proposed to deal with the problem of vanishing gradients that arises in optimizing deep networks.
- Depth has its limitations since its interplay with generalization and optimization is not well understood.