CS4803DL/7643: Deep Learning
Spring 2019
Problem-Set 2 – Architecture Theory

Instructor: Zsolt Kira
TAs: Min-Hung (Steve) Chen, Miao Liu, Anishi Mehta, Sreenivasan Angarai Chandrasekar
Discussions: https://piazza.com/gatech/spring2019/cs7643a

Due: Sunday, March 3, 11:55pm

Instructions

1. We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   - Each problem/sub-problem should be on one or more pages. Anyone who violates this policy will receive an additional penalty on the grade.
   - When submitting to Gradescope, make sure to mark which page corresponds to each problem/sub-problem.
   - Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions. Please read https://web.stanford.edu/class/stats200/gradescope_tips.pdf for additional information on submitting to Gradescope.
   - This portion (PS2) counts 8% of your total grade

2. LaTeX'd solutions are strongly encouraged (solution template available at cc.gatech.edu/classes/AY2019/cs7643_spring/assets/sol2.tex), but scanned handwritten copies are acceptable. Hard copies are **not** accepted.

3. We generally encourage you to collaborate with other students.

   You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and *not* as a group activity. Please list the students you collaborated with.

This assignment introduces you to some theoretical results that address which neural networks can represent what. It walks you through proving some simplified versions of more general results. In particular, this assignment focuses on piecewise linear neural networks, which are the most common type at the moment. The general strategy will be to construct a neural network that has the desired properties by choosing appropriate sets of weights.

1 Logic and XOR

1. **[4 points]** Implement AND and OR for pairs of binary inputs using a single linear threshold neuron with weights $\mathbf{w} \in \mathbb{R}^2$, bias $b \in \mathbb{R}$, and $\mathbf{x} \in \{0, 1\}^2$:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \\ 0 & \text{if } \mathbf{w}^T\mathbf{x} + b < 0 \end{cases} \tag{1}$$

   That is, find $\mathbf{w}_{\texttt{AND}}$ and $b_{\texttt{AND}}$ such that

x_1	x_2	$f_{\texttt{AND}}(\mathbf{x})$
0	0	0
0	1	0
1	0	0
1	1	1

   Also find $\mathbf{w}_{\texttt{OR}}$ and $b_{\texttt{OR}}$ such that

x_1	x_2	$f_{\texttt{OR}}(\mathbf{x})$
0	0	0
0	1	1
1	0	1
1	1	1

2. **[4 points]** Consider the XOR function

x_1	x_2	$f_{\texttt{XOR}}(x)$
0	0	0
0	1	1
1	0	1
1	1	0

   XOR can NOT be represented using one linear model with the same form as (1), but it can be done using two layers of (1). Please prove it and find $\mathbf{w}^1_{\texttt{XOR}}$, $b^1_{\texttt{XOR}}$, $\mathbf{w}^2_{\texttt{XOR}}$, and $b^2_{\texttt{XOR}}$ (1 and 2 are layer numbers).

   Note: The answers are not unique. You can get full credits as long as your answers fit the truth table and you explain the details instead of only providing the final answers.

2 Piecewise Linearity

Consider a specific 2 hidden layer ReLU network with inputs $x \in \mathbb{R}$, 1 dimensional outputs, and 2 neurons per hidden layer. This function is given by

$$h(x) = W^{(3)} \max\{0, W^{(2)} \max\{0, W^{(1)}x + \mathbf{b}^{(1)}\} + b^{(2)}\} + b^{(3)} \tag{2}$$

with weights:

$$W^{(1)} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \tag{3}$$

$$b^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4}$$

$$W^{(2)} = \begin{bmatrix} 1 & 1 \\ 0.5 & 1 \end{bmatrix} \tag{5}$$

$$b^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6}$$

$$W^{(3)} = \begin{bmatrix} 1 & 0.5 \end{bmatrix} \tag{7}$$

$$b^{(3)} = 1 \tag{8}$$

An interesting property of networks with piecewise linear activations like the ReLU is that on the whole they compute piecewise linear functions. For each of the following points give the weight $W \in \mathbb{R}$ and bias $b \in \mathbb{R}$ (report the numerical values) which computes such that $Wx + b = h(x)$. Also compute the gradient $\frac{dh}{dx}$ evaluated at the given point.

1. **[2 points]**

$$x = 1 \tag{9}$$

2. **[2 points]**

$$x = -1 \tag{10}$$

3. **[2 points]**

$$x = -0.5 \tag{11}$$

3 Depth - Composing Linear Pieces

Now we'll turn to a more recent result that highlights the *Deep* in Deep Learning. Depth (composing more functions) results in a favorable combinatorial explosion in the "number of things that a neural net can represent". For example, to classify a cat it seems useful to first find parts of a cat: eyes, ears, tail, fur, *etc*. The function which computes a probability of cat presence should be a function of these components because this allows everything you learn about eyes to generalize to all instances of eyes instead of just a single instance. Below you will detail one formalizable sense of this combinatorial explosion for a particular class of piecewise linear networks.
Consider $y = \sigma(x) = |x|$ for scalar $x \in \mathscr{X} \subseteq \mathbb{R}$ and $y \in \mathscr{Y} \subseteq \mathbb{R}$ (Fig. 1). It has one linear region on $x < 0$ and another on $x > 0$ and the activation identifies these regions, mapping both of them to

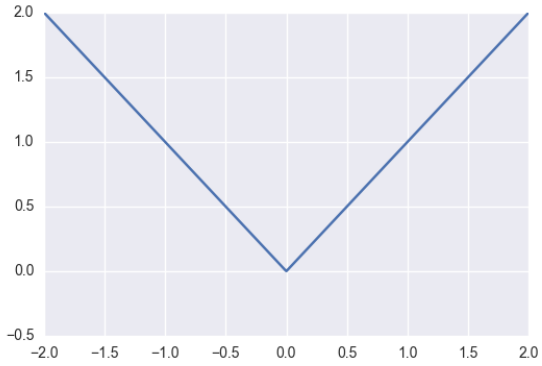

Figure 1

$y > 0$. More precisely, *for each linear region of the input*, $\sigma(\cdot)$ is a bijection. There is a mapping to and from the output space and the corresponding input space. However, given an output y, it's impossible to tell which linear region of the input it came from, thus $\sigma(\cdot)$ *identifies* (maps on top of each other) the two linear regions of its input. This is the crucial definition because when a function identifies multiple regions of its domain that means any subsequent computation applies to all of those regions. When these regions come from an input space like the space of images, functions which identify many regions where different images might fall (*e.g.*, slightly different images of a cat) automatically transfer what they learn about a particular cat to cats in the other regions.
More formally, we will say that $\sigma(\cdot)$ identifies a set of M disjoint input regions $\mathscr{R} = \{R_1, \ldots, R_M\}$ (*e.g.*, $\mathscr{R} = \{(-1, 0), (0, 1)\}$) with $R_i \subseteq \mathscr{X}$ onto one output region $O \subseteq \mathscr{Y}$ (*e.g.*, $(0, 1)$) if for all $R_i \in \mathscr{R}$ there is a bijection from R_i to O. [1]

1. **[6 points](Extra Credit for 4803; Regular Credit for 7643)** Start by applying the above notion of identified regions to linear regions of one layer of a particular neural net that uses absolute value functions as activations. Let $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^d$ [2], and pick weights $W^{(1)} \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b}^{(1)} \in \mathbb{R}^d$ as follows:

$$W_{ij}^{(1)} = \begin{cases} 2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{12}$$

$$b_i^{(1)} = -1 \tag{13}$$

Then one layer of a neural net with absolute value activation functions is given by

$$f_1(\mathbf{x}) = |W^{(1)}\mathbf{x} + \mathbf{b}| \tag{14}$$

Note that this is an absolute value function applied piecewise and not a norm.

How many regions of the input are identified onto $O = (0, 1)^d$ by $f_1(\cdot)$? The answer is 2^d. Prove it. [3]

Feel free to use the conclusion in Q3.1 as a lemma for following questions

[1] Recall that a bijection from X to Y is a function $\mu : X \to Y$ such that for all $y \in Y$ there exists a **unique** $x \in X$ with $\mu(x) = y$.

[2] Outputs are in some feature space, not a label space. Normally a linear classifier would be placed on top of what we are here calling $\mathbf{y}$.

[3] Absolute value activations are chosen to make the problem simpler, but a similar result holds for ReLU units. Also, O could be the positive orthant (unbounded above).

2. **[4 points]** Next consider what happens when two of these functions are composed. Suppose g identifies n_g regions of $(0,1)^d$ onto $(0,1)^d$ and f identifies n_f regions of $(0,1)^d$ onto $(0,1)^d$. How many regions of its input does $f \circ g(\cdot)$ identify onto $(0,1)^d$?

3. **[4 points]** Finally consider a series of L layers identical to the one in question 3.1.

$$\mathbf{h}_1 = |W_1\mathbf{x} + \mathbf{b}_1| \tag{15}$$
$$\mathbf{h}_2 = |W_2\mathbf{h}_1 + \mathbf{b}_2| \tag{16}$$
$$\vdots \tag{17}$$
$$\mathbf{h}_L = |W_L\mathbf{h}_{L-1} + \mathbf{b}_L| \tag{18}$$

Let $\mathbf{x} \in (0,1)^d$ and $f(\mathbf{x}) = \mathbf{h}_L$. Note that each $\mathbf{h}_i$ is *implicitly* a function of $\mathbf{x}$. Show that $f(\mathbf{x})$ identifies 2^{Ld} regions of its input.

4 Conclusion to Theory Part

Now compare the number of identified regions for an L layer net to that of an $L-1$ layer net. The L layer net can separate its input space into 2^d more linear regions than the $L-1$ layer net. On the other hand, the number of parameters and the amount of computation time grows linearly in the number of layers. In this very particular sense (which doesn't always align well with practice) deeper is better.

To summarize this problem set, you've shown a number of results about the representation power of different neural net architectures. First, neural nets (even single neurons) can represent logical operations. Second, neural nets we use today compute piecewise linear functions of their input. Third, the representation power of neural nets increases exponentially with the number of layers.

The point of the exercise was to convey intuition that removes some of the magic from neural nets representations. Specifically, neural nets can decompose problems logically, and piecewise linear functions can be surprisingly powerful.