6.390 Introduction to Machine Learning

Recitation Week #6

Issued March 13, 2022

1. Nori thinks about reLU units and wonders whether there is a better alternative for an activation function, and decides to explore the LUre function, defined as:

$$f_{\text{LUre}}(z) = \min(z, 0).$$

(a) Sketch $f_{\text{reLU}}(z)$ and $f_{\text{LUre}}(z)$.

(b) What is the derivative of this function, $\frac{df_{\text{LUre}}(z)}{dz}$?

(c) Nori's friend Ori thinks this is cool and suggests making a neural network with *two* activation functions per layer, in particular,

$$a^l = f_{\text{LUre}}(f_{\text{reLU}}(z^l)).$$

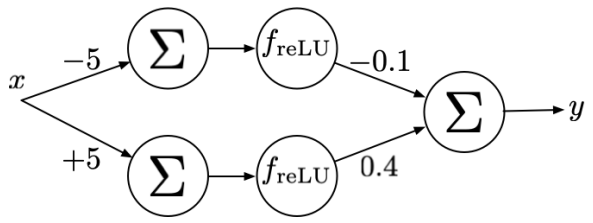
Explain what effect this will have on the network.

(d) Nori's other friend Dori thinks we should try this trick with two reLUs, so that,

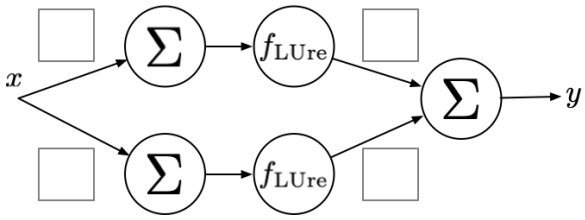
$$a^l = f_{\text{reLU}}(f_{\text{reLU}}(z^l)).$$

Explain what effect this will have on the network.

(e) Nori finds a neural network trained by her nemesis Smori that takes a single-dimensional input ($d = 1$) and looks like this (with no constant offsets into the summation):

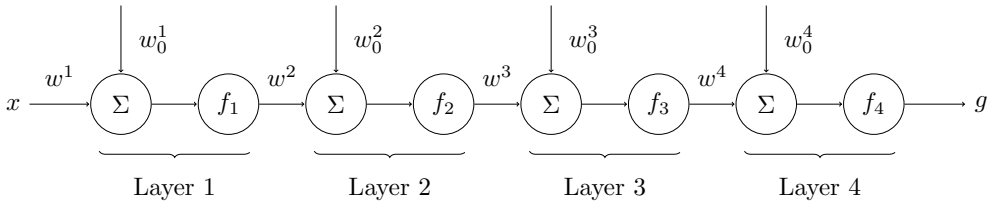


She sees that it computes $\hat{y} = -0.1 f_{\text{reLU}}(-5x) + 0.4 f_{\text{reLU}}(5x)$ and is very curious to see if she can replace those reLU activation units with her own LUre's. Please help her find another neural network that computes exactly the same function as the one above (that is, maps any input x to the same output as the original one). Provide a set of weights that achieves this in the boxes on the diagram below.



(f) Is the resulting set of weights found by Nori in (d) unique? Would it be possible to create an equivalent network if Smori were to include offsets at each neuron? Why or why not?

2. Kim constructs a fully connected deep neural network with 4 layers, pictured in the figure below. He uses ReLU activation functions for all hidden layers, denoted by f_1, f_2, f_3 in the figure, and an identity activation $f(z) = z$ for the output layer, denoted by f_4, and a squared-error loss for training. The ReLU activation function is implemented as $\text{ReLU}(z) = \max(0, z)$, with $\partial\text{ReLU}(z)/\partial z = 1$ if $z > 0$, and 0 otherwise. Kim has a data set $\mathcal{D}_n = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$, where each $x^{(i)}$ is a 1-dimensional feature and $y^{(i)}$ is a 1-dimensional label.



Consider the following, which will help us represent how the neural network is operating:

$$a^0 = x, \quad z^l = w^l a^{l-1} + w_0^l, \quad a^l = f_l(z^l), \quad g = a^4.$$

The weights are initialized as follows:

$$w_0^1 = -1,\ w^1 = 3,\ w_0^2 = 1,\ w^2 = 4,\ w_0^3 = -5,\ w^3 = 1,\ w_0^4 = 1,\ w^4 = 1.$$

(a) Before training, Kim is curious about the output of his network as initialized. What will Kim observe at the output of the neural network when he provides the feature, $x^{(1)} = 1$? For each layer l, compute the values of a^l, z^l by means of a *forward pass*.

	$a^0 =$
$z^1 =$	$a^1 =$
$z^2 =$	$a^2 =$
$z^3 =$	$a^3 =$
$z^4 =$	$a^4 =$

(b) Following the above construction, Kim wants to derive the formula for back-propagation. He uses the squared-error loss function, $\mathcal{L}(g^{(i)}, y^{(i)}) = (g^{(i)} - y^{(i)})^2$, where $g^{(i)} = \text{NN}(x^{(i)}; W)$, and W collects all of the weights and offsets across all of the layers. Kim derives the gradient of the loss function with respect to weight w^1 as:

$$\frac{\partial \mathcal{L}(g,y)}{\partial w^1} = \frac{\partial \mathcal{L}(g,y)}{\partial g} \cdot \frac{\partial g}{\partial z^4} \cdot \frac{\partial z^4}{\partial a^3} \cdot \frac{\partial a^3}{\partial z^3} \cdot \frac{\partial z^3}{\partial a^2} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial z^2}{\partial a^1} \cdot \frac{\partial a^1}{\partial z^1} \cdot \frac{\partial z^1}{\partial w^1}.$$

Provide equations for each of the factors in the equation above:

$\frac{\partial z^1}{\partial w^1} =$	$\frac{\partial a^1}{\partial z^1} =$	$\frac{\partial z^2}{\partial a^1} =$
$\frac{\partial a^2}{\partial z^2} =$	$\frac{\partial z^3}{\partial a^2} =$	$\frac{\partial a^3}{\partial z^3} =$
$\frac{\partial z^4}{\partial a^3} =$	$\frac{\partial g}{\partial z^4} =$	$\frac{\partial \mathcal{L}}{\partial g} =$

(c) Now, we are well equiped to compute a gradient descent step for updating the weight w^1 through backpropagation. Using the formula,

$$w^1 = w^1 - \eta \frac{\partial \mathcal{L}(g,y)}{\partial w^1},$$

and the components that you found in parts (a) and (b), calculate one gradient descent update for the training data point $(x^{(1)}, y^{(1)}) = (1, 2)$, and a step size $\eta = 0.1$.

(d) Kim next looks to find the gradient with respect to the offset to the first neuron, w_0^1. Write out the equation for $\frac{\partial \mathcal{L}(g,y)}{\partial w_0^1}$, and identify which factors are shared with the equation for $\frac{\partial \mathcal{L}(g,y)}{\partial w^1}$.