EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade

Discussion 2

1. Visualizing Derivatives

Consider a simple neural network that takes a scalar real input, has 1 hidden layer with k units in it and a ReLU nonlinearity for those units, and an output linear (affine) layer.

We can algebraically write any function that it represents as

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

Where $x, y \in \mathbb{R}$, $W^{(1)} \in \mathbb{R}^{k\times 1}$, $W^{(2)} \in \mathbb{R}^{1\times k}$, and $\mathbf{b}^{(1)} \in \mathbb{R}^{k\times 1}$, and $b^{(2)} \in \mathbb{R}$. The superscripts are indices, not exponents and the $\max$ given two vector arguments applies the max on corresponding pairs and returns a vector. You may assume values for b's and w's, for example, the solutions will use these values for the plots: $w_1^{(1)} = 2, w_2^{(1)} = 2, b_1^{(1)} = -1, b_2^{(1)} = -2, w_1^{(2)} = 0.5, w_2^{(2)} = 2.0$.

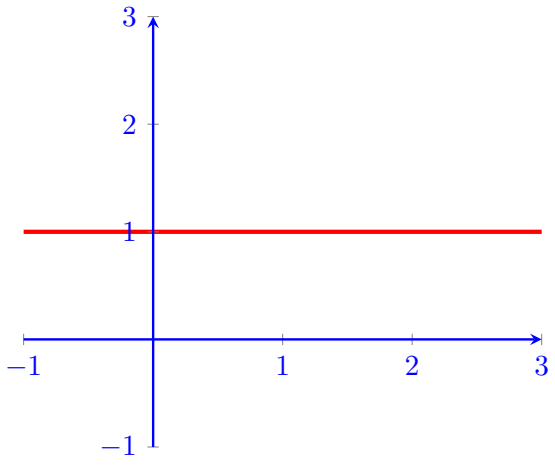
For each part, **calculate the partial derivative and sketch a small representative plot of the derivative as a function of** x. Make sure to clearly label any discontinuities, kinks, and slopes of segments. The subscript i refers to the i-th element of a vector.

Solution: Note that the numeric values on the axes, except for part (a), are **not** indicative of where the non-linearity/discontinuity happens, so please focus on the shape of the function and not the numbers. The numbers are by-products generated by the online graphing calculator. All the critical points in the following subparts happen at $x = \frac{-b_i^{(1)}}{W_i^{(1)}}$.

(a) $\frac{\partial y}{\partial b^{(2)}}(x)$

Solution:

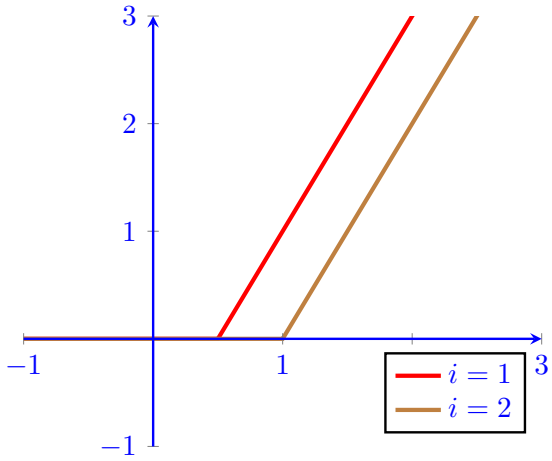
$$\frac{\partial y}{\partial b^{(2)}} = 1$$



(b) $\frac{\partial y}{\partial w_i^{(2)}}(x)$

Solution:

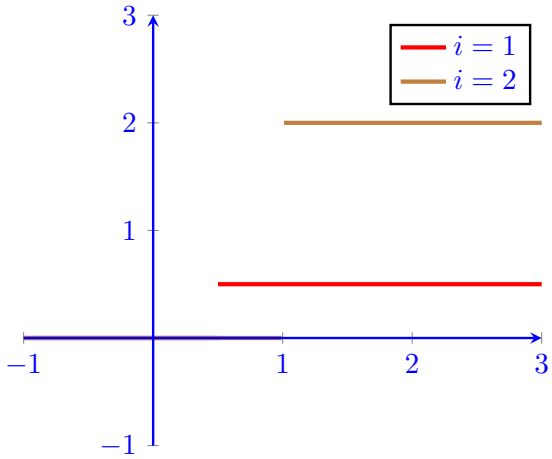
$$\frac{\partial y}{\partial w_i^{(2)}} = \max(0, w_i^{(1)}x + b_i^{(1)})$$



(c) $\frac{\partial y}{\partial b_i^{(1)}}(x)$

Solution:

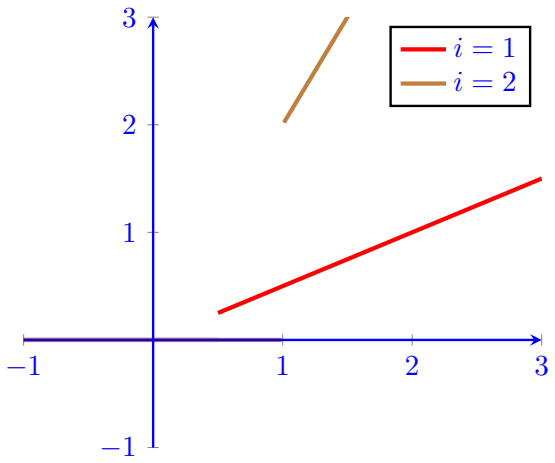
$$\frac{\partial y}{\partial b_i^{(1)}} = \begin{cases} w_i^{(2)}, \text{if } w^{(1)}x + b^{(1)} > 0 \\ 0, \text{if } w^{(1)}x + b^{(1)} < 0 \end{cases}$$



(d) $\frac{\partial y}{\partial w_i^{(1)}}(x)$

Solution:

$$\frac{\partial y}{\partial w_i^{(1)}} = \begin{cases} w_i^{(2)}x, \text{if } w^{(1)}x + b^{(1)} > 0 \\ 0, \text{if } w^{(1)}x + b^{(1)} < 0 \end{cases}$$



(e) When you fine-tune your network, you observe a coupled motion in your parameters as gradient descent adjusts them. In particular, whenever the bias $b_1^{(1)}$ on the first hidden unit moves up by 0.01 it seems that the weight $w_2^{(1)}$ on the second hidden unit also moves up by 0.02. Draw a 1-dimensional synthetic feature (function of x) that represents this observed regularity. Sketch a small representative plot of this synthetic feature.

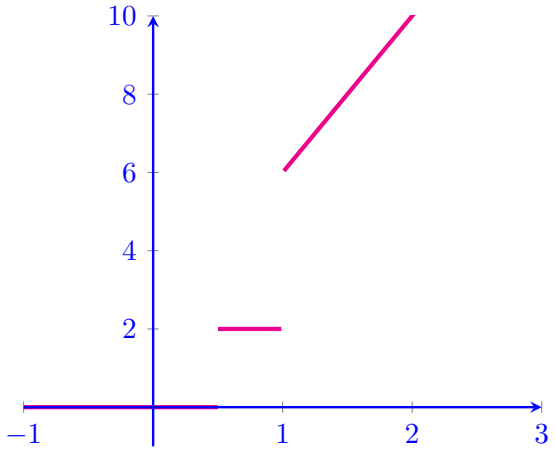
Hint: Fix all the parameters in the network except for $b_1^{(1)}$ and $w_2^{(1)}$ and consider how the output will change due to small changes in these two parameters. We can approximate this with the following:

$$\theta = \begin{bmatrix} w_2^{(1)} & b_1^{(1)} \end{bmatrix} \tag{1}$$

$$y = f_{\theta_0+\Delta\theta}(x) \approx f_{\theta_0}(x) + \frac{\partial f}{\partial \theta}(x)\Delta\theta. \tag{2}$$

Solution: Our new synthetic feature that captures this coupled motion is

$$s(x) = 2\frac{\partial y}{\partial w_2^{(1)}} + \frac{\partial y}{\partial b_1^{(1)}} = \begin{cases} 2w_2^{(2)}x + w_1^{(2)}, \text{ if } x > 1 \\ w_1^{(2)}, \text{ if } 0.5 < x < 1 \\ 0, \text{ if } x < 0.5 \end{cases} \tag{3}$$

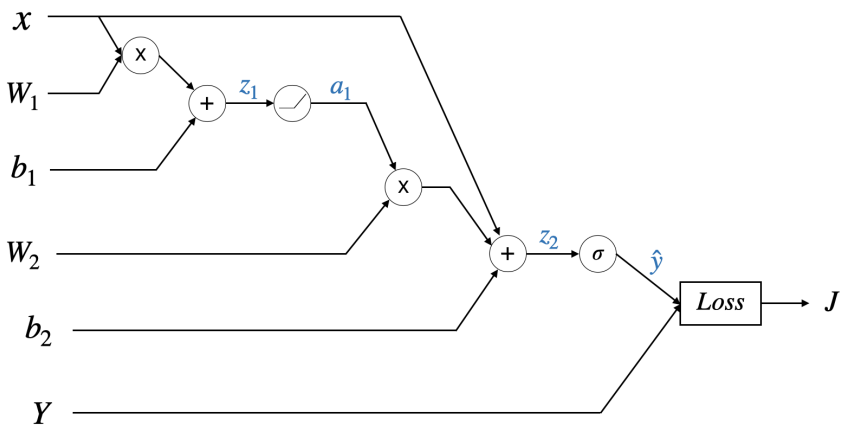


2. Computation Graph Review

One of the reasons why modern neural networks are growing much larger than ten years ago is due to the advent of automatic differentiation softwares, also known as *Autodiff*. Autodiff softwares can compute the gradients of any differentiable function by constructing a *computation graph* of such functions, which allows researchers to simply focus on building models with effective information propagation and not to worry about the complex back prop.

Here is the computation graph of an one hidden layer MLP with *skip connection*, which basically means adding the input to the output of some later layers (this is a commonly used design that we will cover in detail later in the course). σ denotes the sigmoid function and J is the final loss. Assume that we use binary cross-entropy as our loss function with a ground truth label y, ie. $loss = -y \ln \hat{y} - (1-y)\ln(1-\hat{y})$.
You may find this quantity useful:

- $\frac{\partial J}{\partial z_2} = \hat{y} - y$



(a) **Express $\hat{y}$ as a function of x, W_1, b_1, W_2 and b_2.**

Solution:

$$\begin{aligned}\hat{y} &= \sigma(z_2) \\ &= \sigma(W_2 a_1 + b_2 + x) \\ &= \sigma(W_2 \cdot ReLU(W_1 x + b_1) + b_2 + x)\end{aligned} \tag{4}$$

(b) **Compute the derivatives $\frac{\partial J}{\partial W_2}$, $\frac{\partial J}{\partial b_2}$.**

Solution:

- $\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial W_2} = (\hat{y} - y) \cdot a_1$
- $\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} = \hat{y} - y$

(c) **Compute the derivatives $\frac{\partial J}{\partial W_1}$, $\frac{\partial J}{\partial b_1}$, $\frac{\partial J}{\partial x}$.**

Solution:

Note how $\frac{\partial J}{\partial x}$ relies on two streams of derivatives.

- $\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial W_1} = (\hat{y} - y) \cdot W_2 \cdot \frac{\partial a_1}{\partial z_1} \cdot x$

- $\frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} = (\hat{y} - y) \cdot W_2 \cdot \frac{\partial a_1}{\partial z_1}$, where

$$\frac{\partial a_1}{\partial z_1} = \begin{cases} 1, \text{if } z_1 > 0 \\ 0, \text{if } z_1 < 0 \end{cases}$$

- $\frac{\partial J}{\partial x} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial x} + \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial x} = (\hat{y} - y) \cdot W_2 \cdot \frac{\partial a_1}{\partial z_1} \cdot W_1 + (\hat{y} - y)$

(d) What **intermediate variables do we need to cache** in this case? **Why is it more efficient to have an additional backward pass** on top of the forward pass for the computation of derivatives?

Solution:

We will need to cache $\hat{y}$, x and all the intermediate variables z_2, z_1, a_1, as we will need them when applying the chain rule.

As we see in part (b) and (c), the downstream derivatives depend on the upstream. Therefore by doing back propagation, we may compute all the parameters' derivatives in a dynamic programming manner according to the computation graph, without having to compute the same partial derivative terms repeatedly.

3. ReLU with different Optimizers

Work through the notebook to explore how the simple neural network with ReLU non-linearities from last discussion learns using different optimizers. The notebook compares SGD, SGD with momentum, and Adam. Notebook url - `https://tinyurl.com/cs182-dis02-code`

Solution: Answers to notebook questions.

- How does the hidden layer width and different optimizers impact the learned function and test error? **Ans:** The larger hidden layer width is, the lower test error we get. Different optimizers result in different performance. Both Adam and SGD with momentum are consistently better than SGD. Generally Adam converges faster than both versions of SGD. In terms of variance (error bar), SGD has the largest variance among different random seeds, and Adam is consistently the most stable one.
- What happens to the elbow locations using different optimizers during training? **Ans:** Converging approximately to the non-linear kinks (for detailed reason, check out discussion 1 notebook). Next, we notice multiple ReLUs converge to the same elbow locations.
- Are the circle dots on the graphs above mean or median? Why not plot the other one? **Ans:** Median. It does not make sense to plot the mean since we are plotting at log scale, and averaging points may yield values that are not well represented on the plot.
- How are error bars computed? Are the upper and lower marks maximum/minimum, standard deviation or something else? Does it make sense to plot standard deviation here? **Ans:** The Upper and lower bound are 25% and 75% percentile respectively. Plotting S.D. does not make sense here again due to the log scale plotting.

Contributors:

- Saagar Sanghavi.
- Anant Sahai.
- Qiyang Li.

- Kevin Li.
- Kumar Krishna Agrawal.
- Naman Jain.

