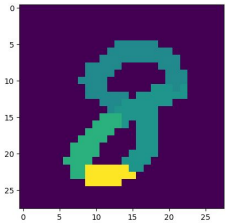
446 Section



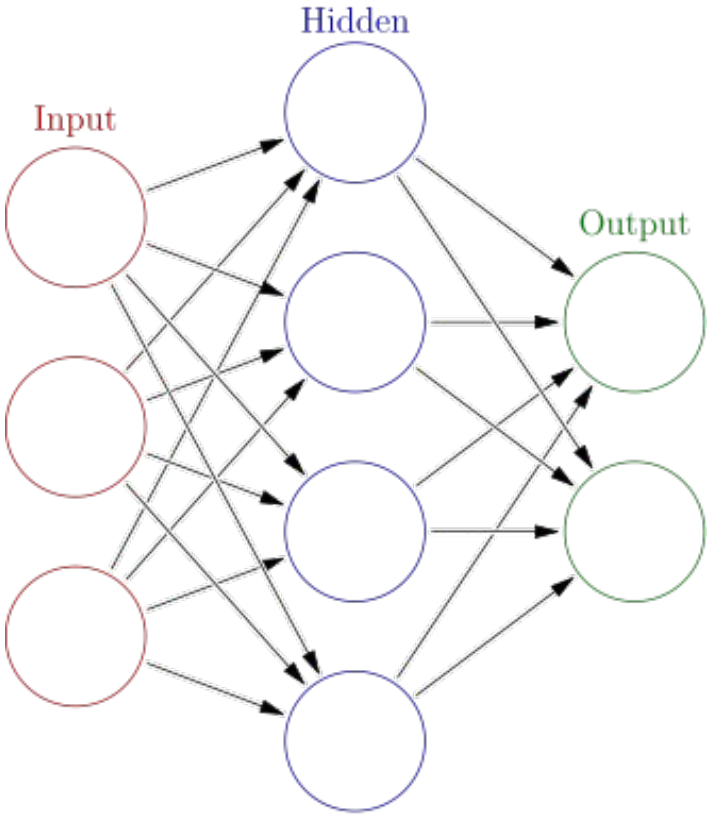
Plans for today!

1. This
2. Reminders
3. Neural Networks
    a. Forward/Backward passes
    b. Weight Initialization

Reminders

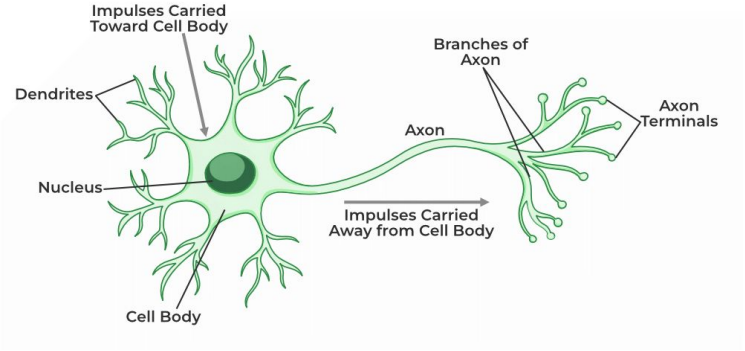
- HW3 ~~was due yesterday~~ now due 5/22
- HW4 was released yesterday, due 6/3
  - Last hw!

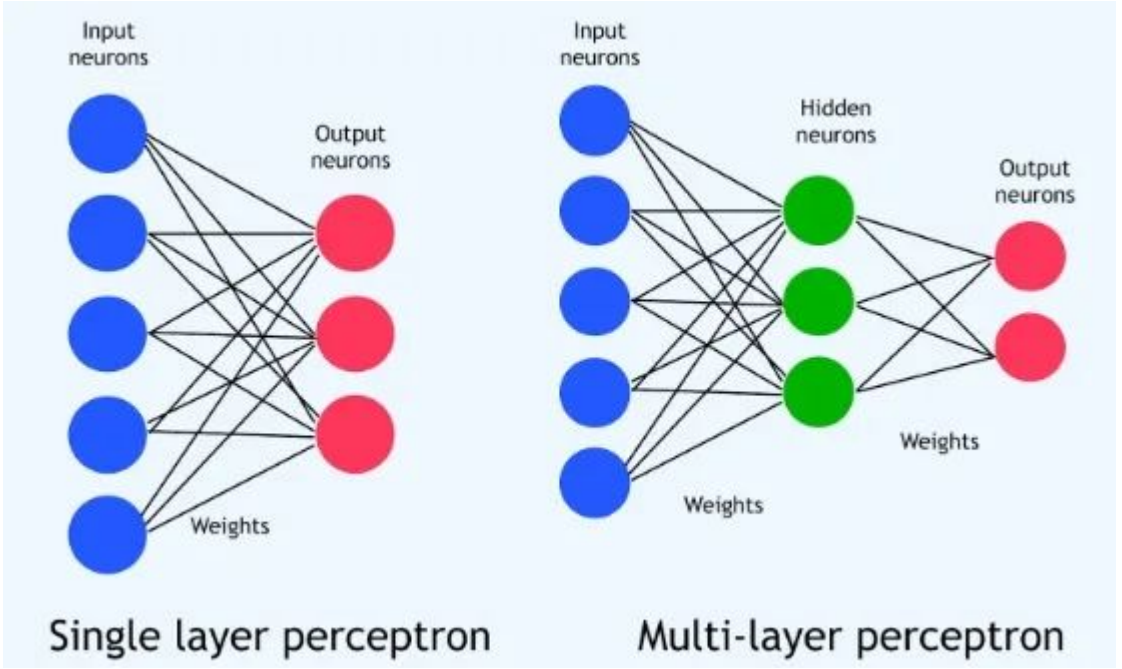
Neural Networks



Forward Pass

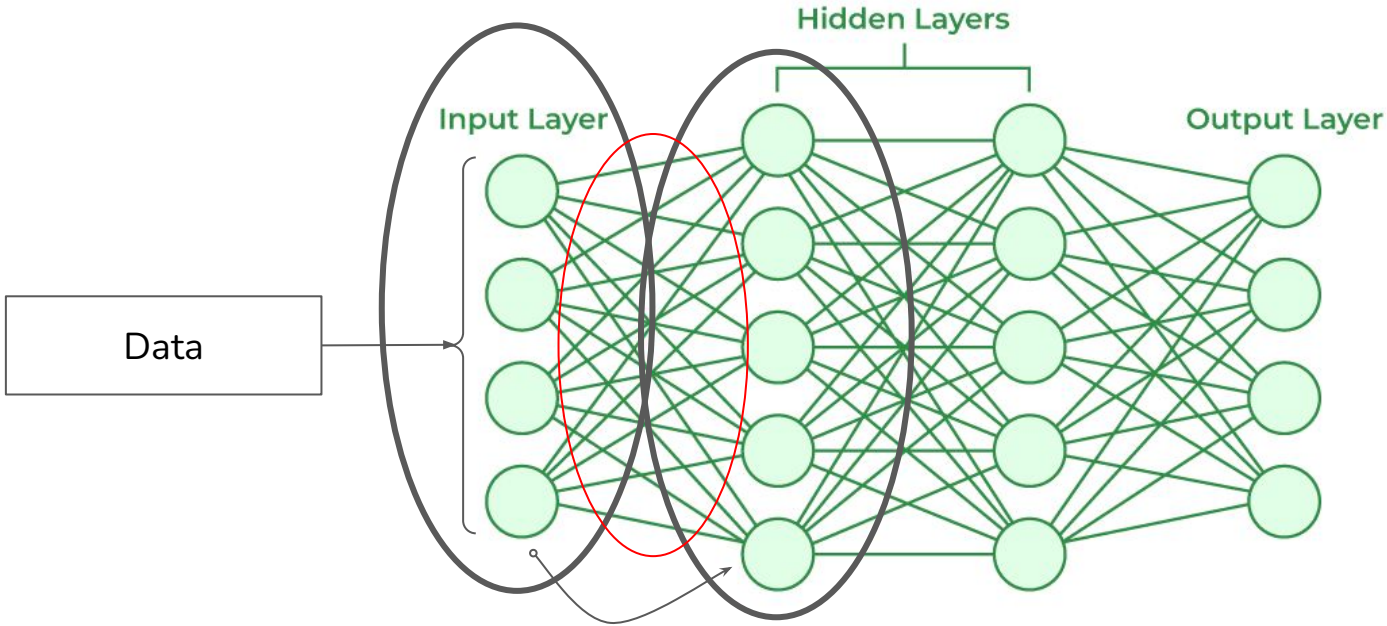
Background: The Perceptron





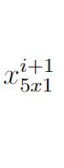
Originally to classify between two categories
(i.e. one output node)

Passing Data Through the Network



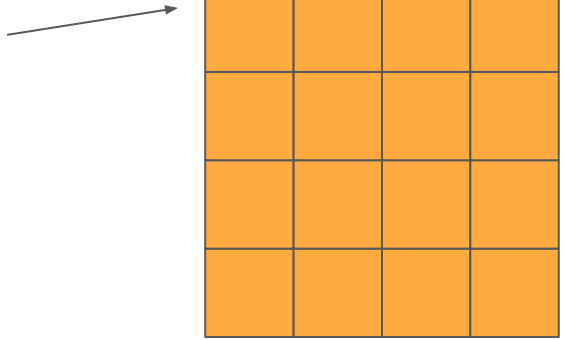
How is this done?

Goal:



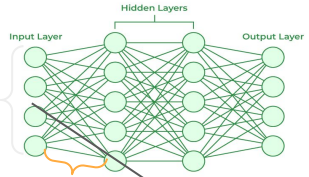
x^{i+1}_{5x1}

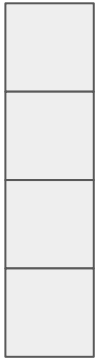
Floats initialized semi-randomly. Learning these = learning the function



W_{5x4}

Weights





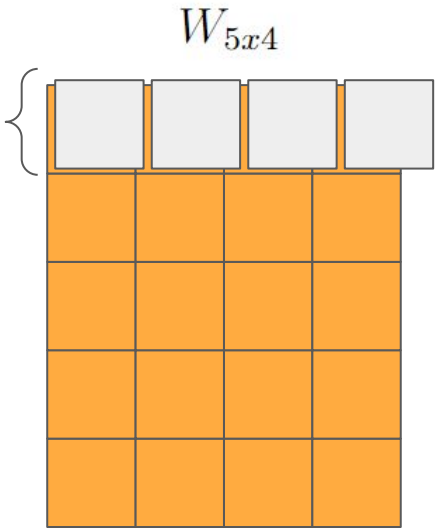
x^{i}_{4x1}

Data

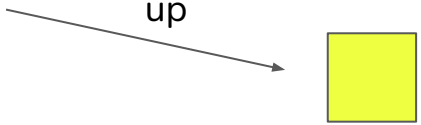
How is this done?

Goal:  x_{5x1}^{i+1}

Multiply each square of the input with the square under it

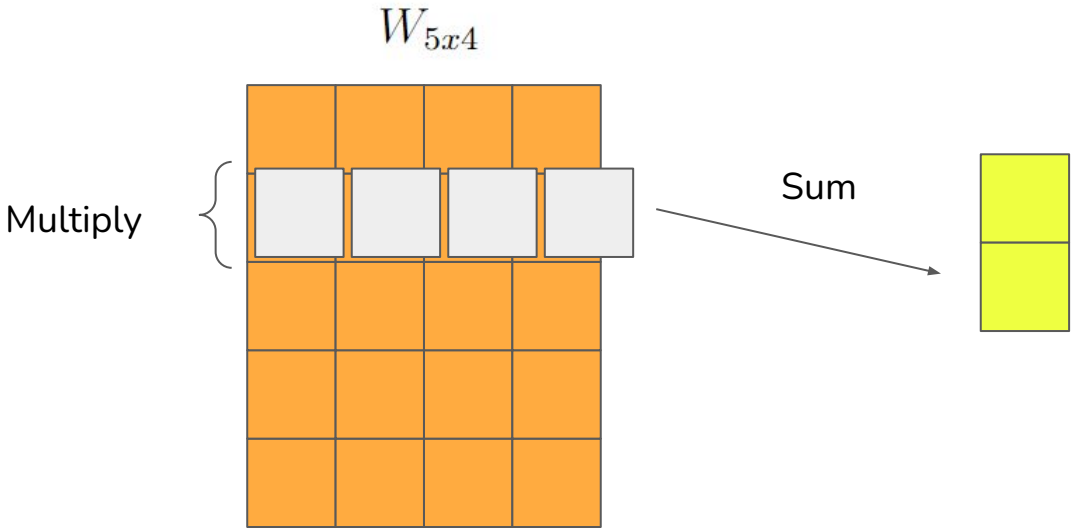


Sum all of those up



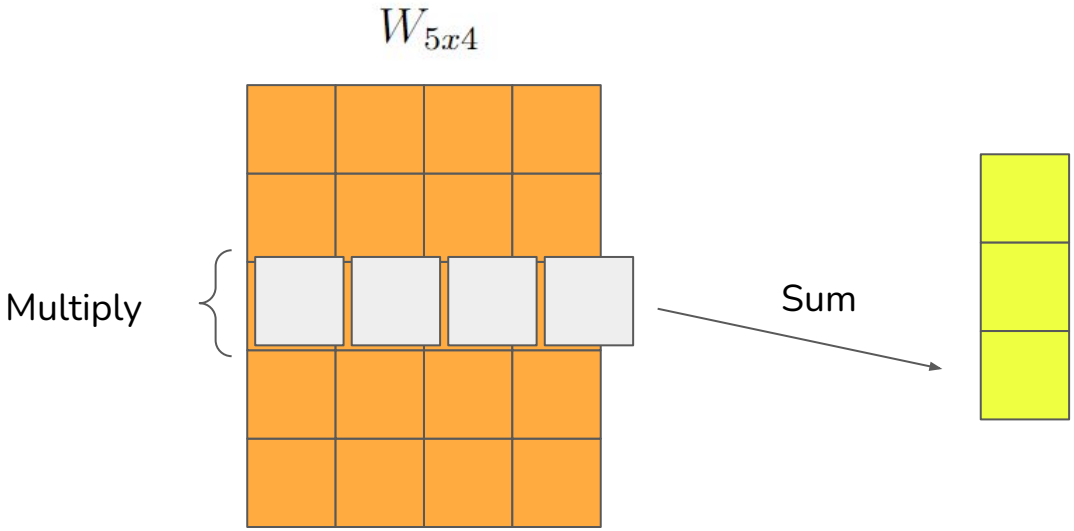
How is this done?

Goal: x_{5x1}^{i+1}



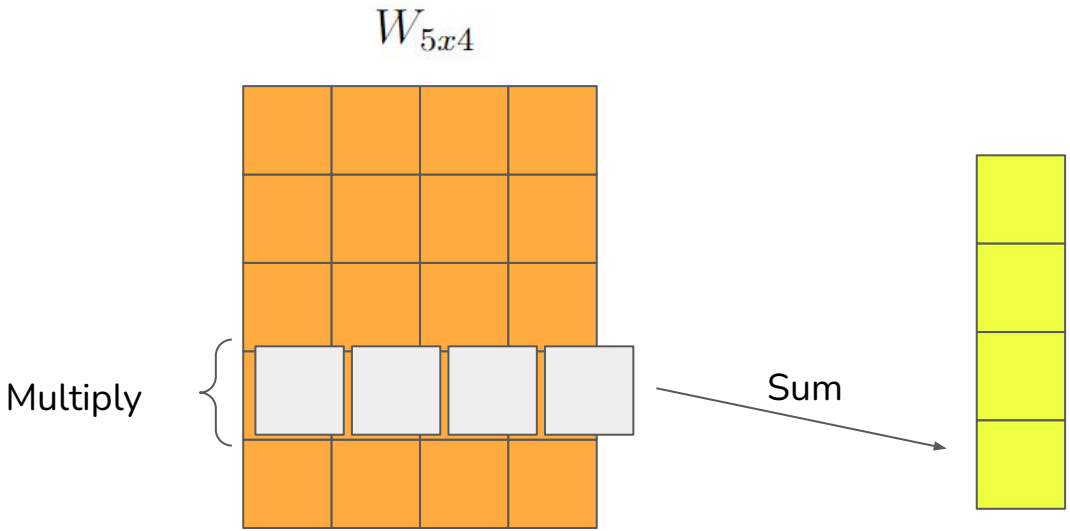
How is this done?

Goal:  x_{5x1}^{i+1}



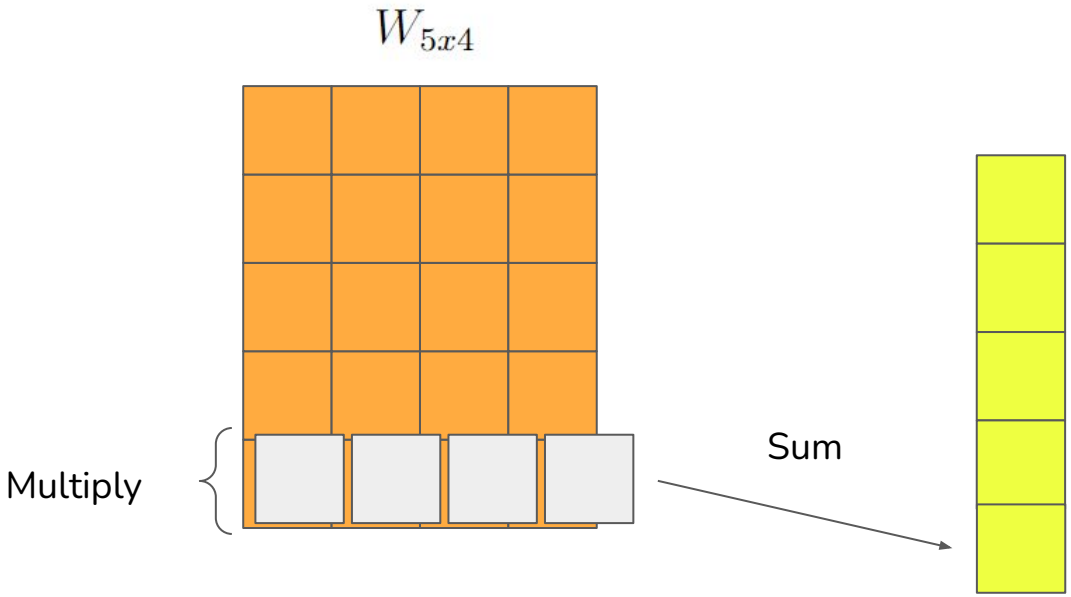
How is this done?

Goal: x_{5x1}^{i+1}



How is this done?

Goal:  x_{5x1}^{i+1}



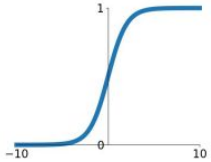
Sigmoid Activation Function

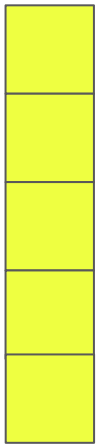
Goal:  x_{5x1}^{i+1}

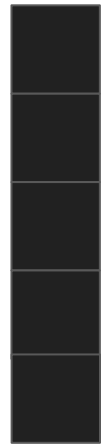
Sigmoid activation function that squishes inputs between 0 and 1!

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



σ 

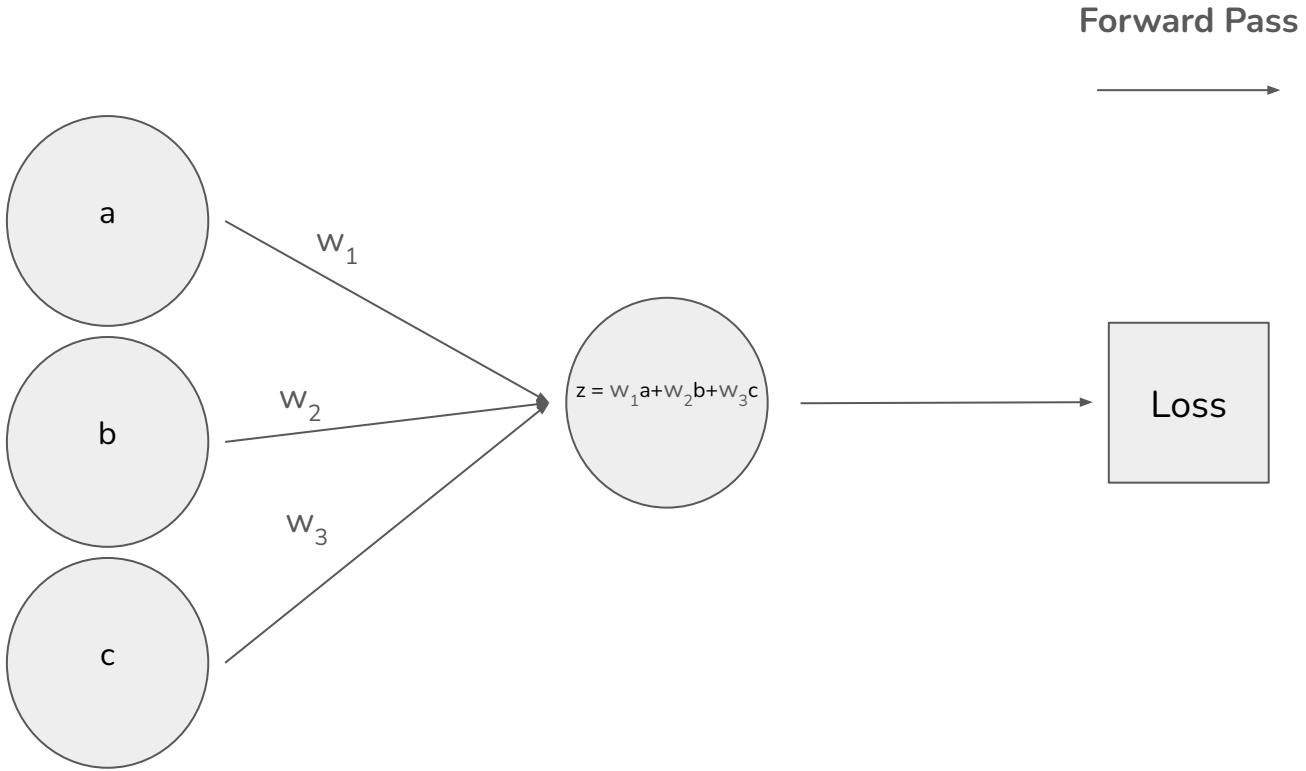


x_{5x1}^{i+1}

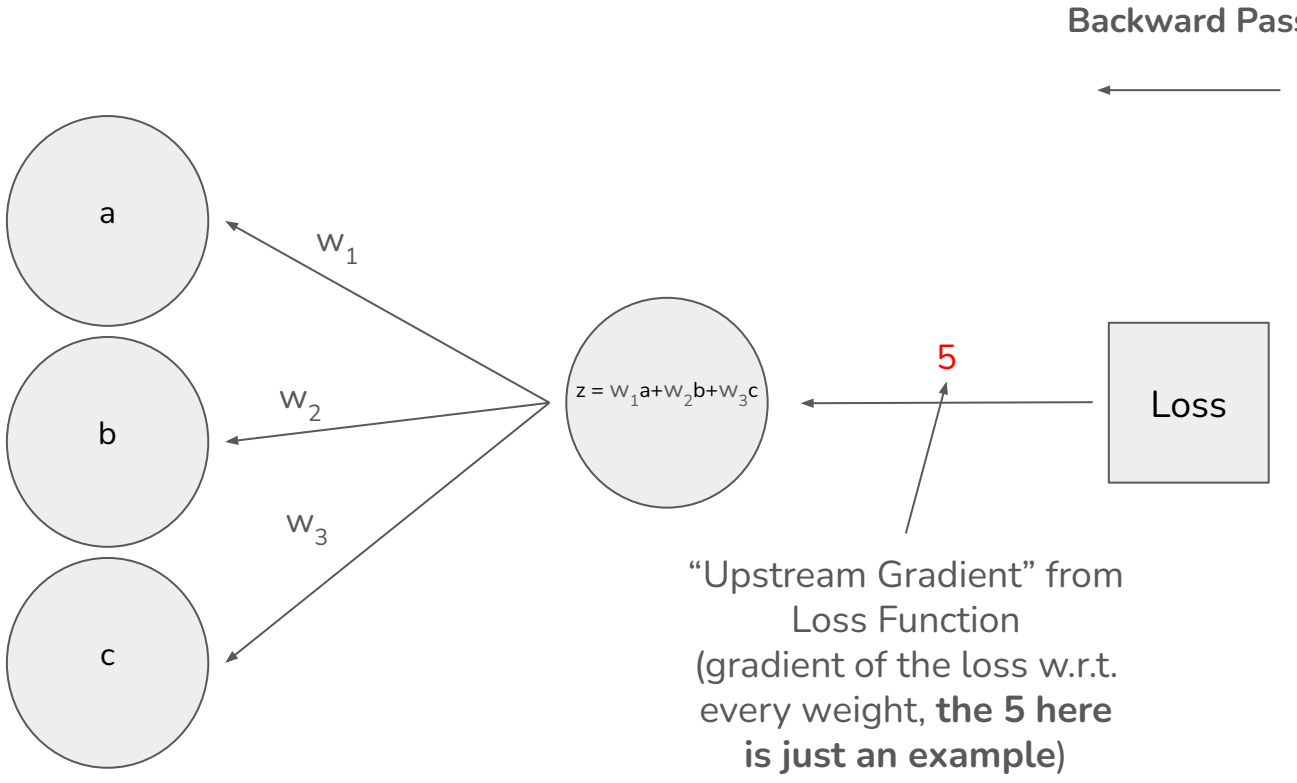
Sigmoid does have some problems though...

Backpropagation

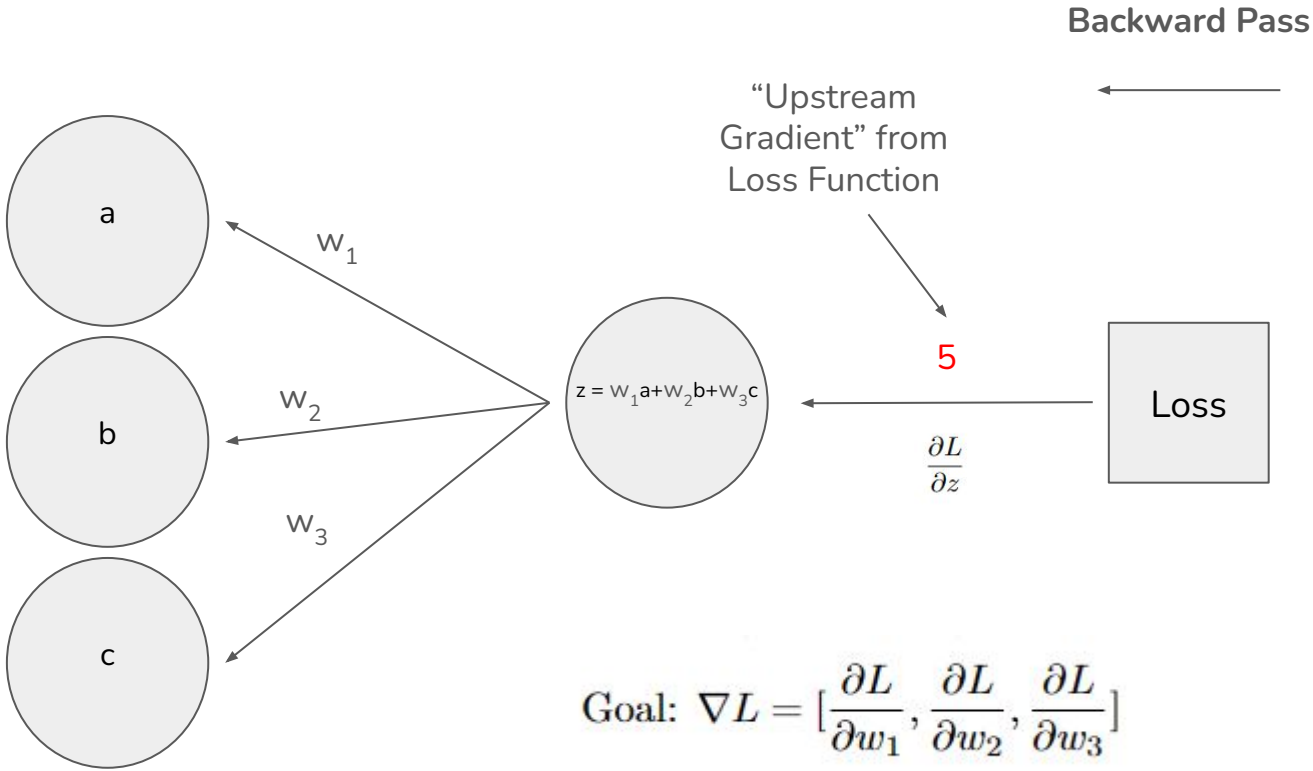
Backpropagation Intuition



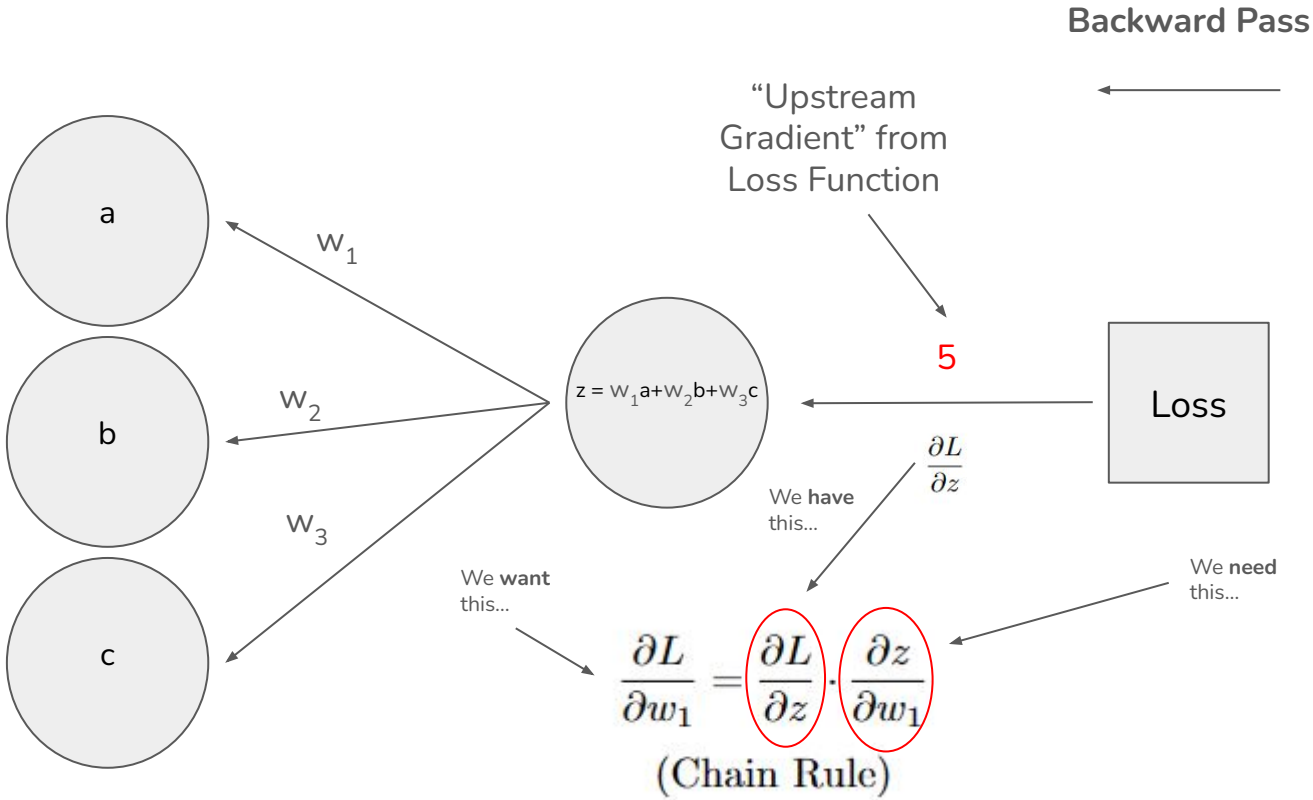
Backpropagation Intuition



Backpropagation Intuition



Backpropagation Intuition



Backpropagation Intuition

$$\frac{\partial}{\partial w_1} z = \frac{\partial}{\partial w_1}(w_1 a + w_2 b + w_3 c)$$

$$\frac{\partial z}{\partial w_1} = a$$

$$\frac{\partial L}{\partial z} = 5$$

$$\frac{\partial L}{\partial w_1} = 5 * a$$

$$\frac{\partial}{\partial w_2} z = \frac{\partial}{\partial w_2}(w_1 a + w_2 b + w_3 c)$$

$$\frac{\partial z}{\partial w_2} = b$$

$$\frac{\partial L}{\partial z} = 5$$

$$\frac{\partial L}{\partial w_2} = 5 * b$$

$$\frac{\partial}{\partial w_3} z = \frac{\partial}{\partial w_3}(w_1 a + w_2 b + w_3 c)$$

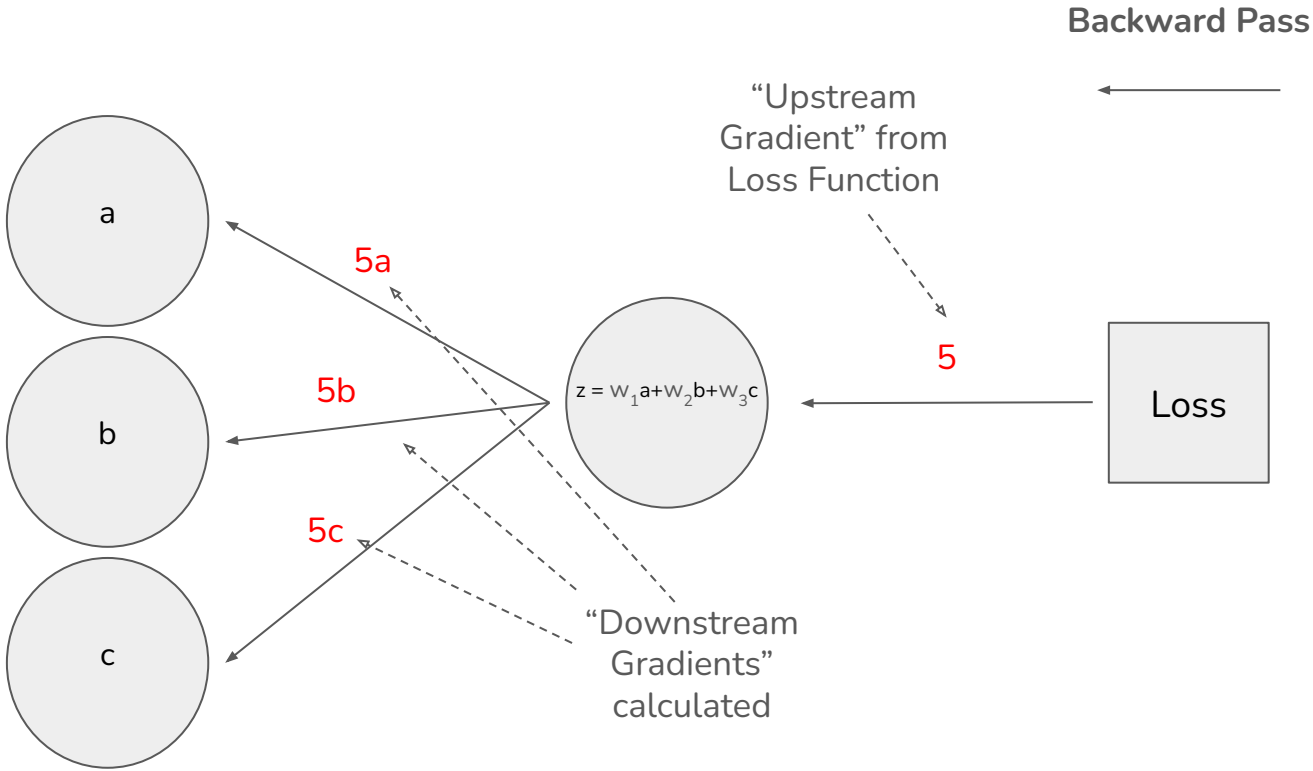
$$\frac{\partial z}{\partial w_3} = c$$

$$\frac{\partial L}{\partial z} = 5$$

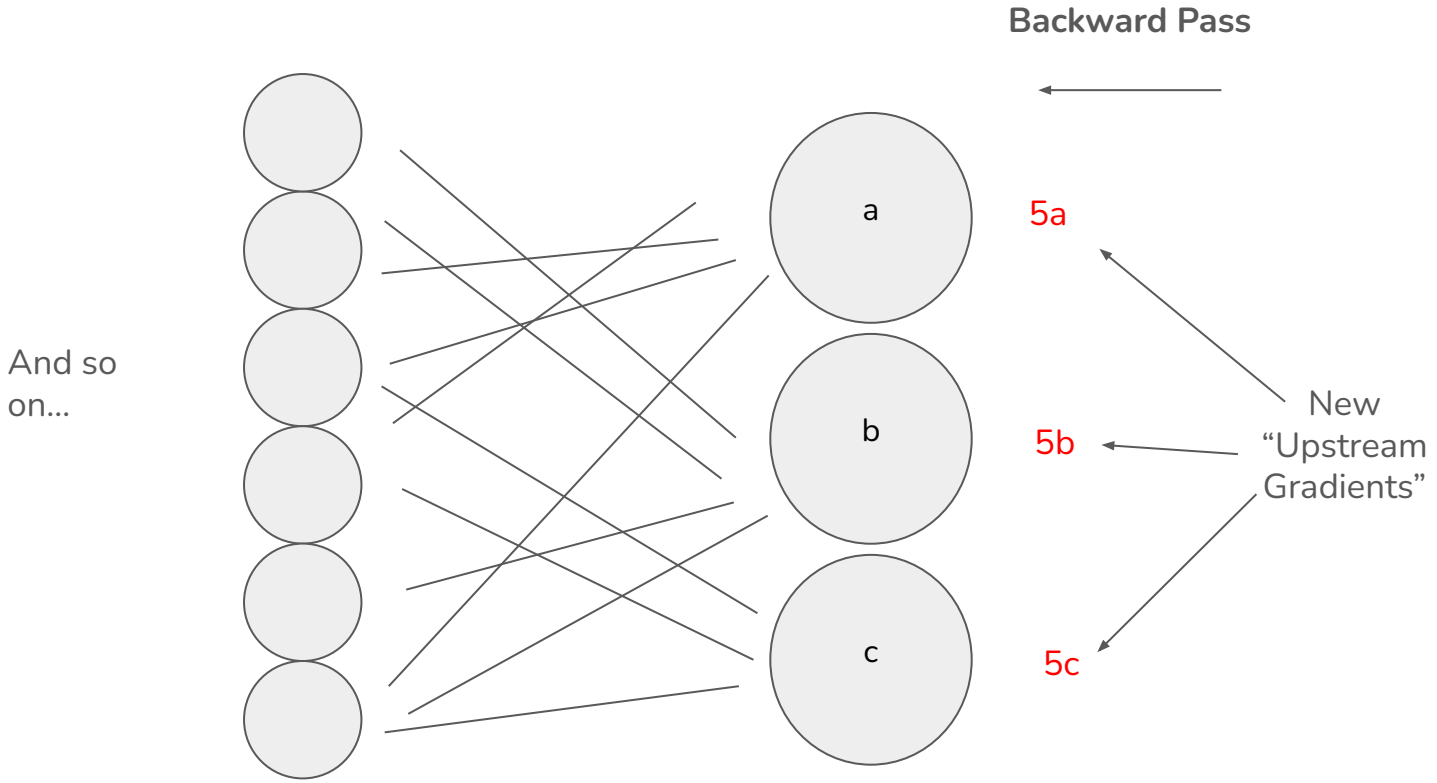
$$\frac{\partial L}{\partial w_3} = 5 * c$$

$$\text{Goal: } \nabla L = [\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}]$$

Backpropagation Intuition

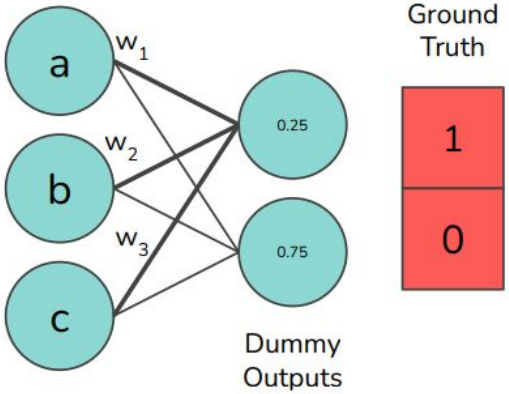


Backpropagation Intuition



Dummy Backpropagation Additional Full Example

Dummy Network



$$\text{Goal: } \nabla L = [\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}]$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_1}$$

$$L = (\hat{y}_i - y_i)^2$$

$$\frac{\partial L}{\partial \hat{y}_i} = 2(\hat{y}_i - y_i)$$

$$\hat{y}_i = aw_1 + bw_2 + cw_3$$

$$\frac{\partial \hat{y}_i}{\partial w_1} = a, \frac{\partial \hat{y}_i}{\partial w_2} = b, \frac{\partial \hat{y}_i}{\partial w_3} = c$$

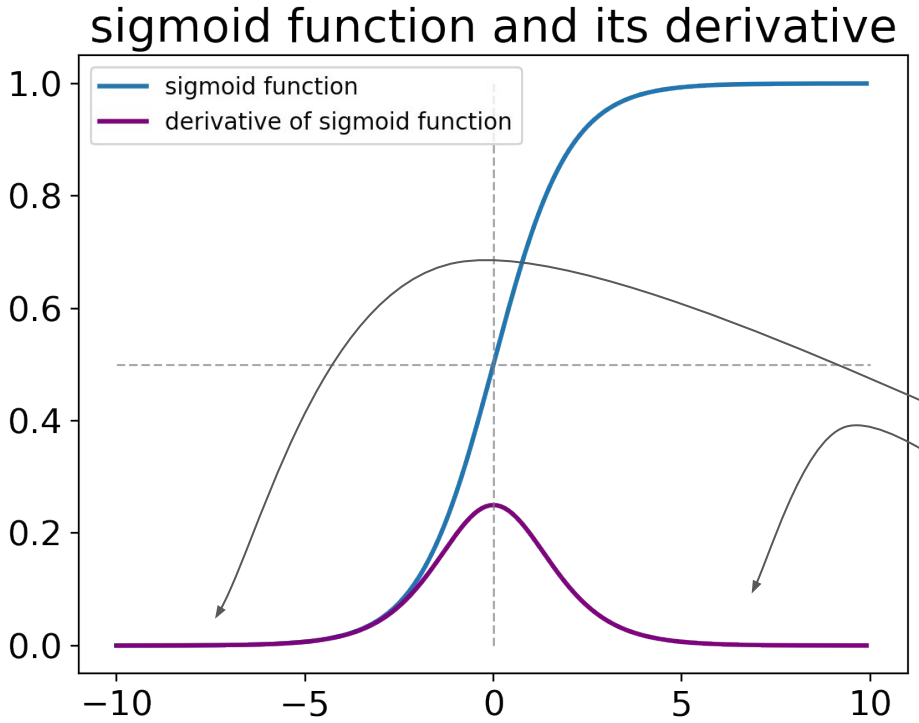
$$\nabla L = [2(\hat{y}_i - y_i) * a, 2(\hat{y}_i - y_i) * b, 2(\hat{y}_i - y_i) * c]$$

$$w_{new} = w_{old} - \lambda \frac{\partial L}{\partial w}$$

Issue with Sigmoid from earlier

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d}{dx}\sigma(x) = \sigma'(x) = \sigma(x)(1 - \sigma(x))$$

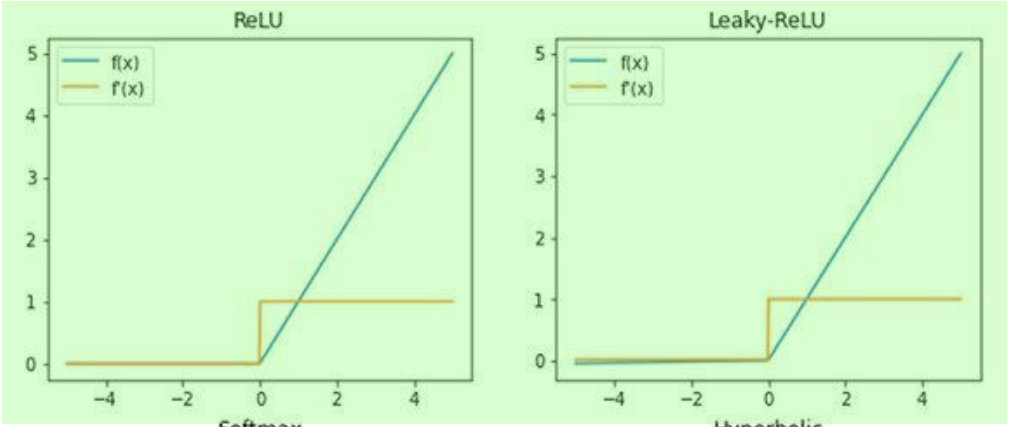


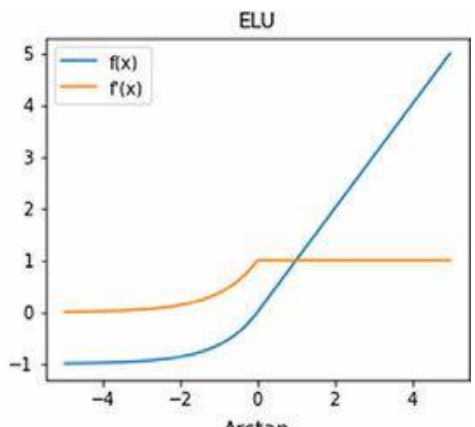
The **derivative of the sigmoid function** is what upstream gradients must pass through to get to layers further back

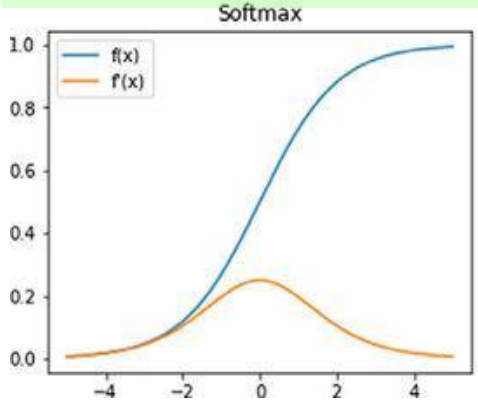
The sigmoid function has these “dead zones” that can kill information flowing backwards!

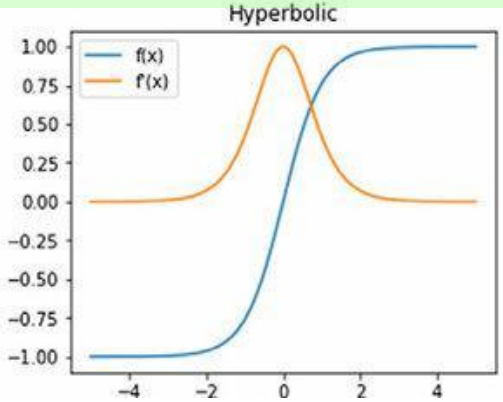
Issue with Sigmoid from earlier

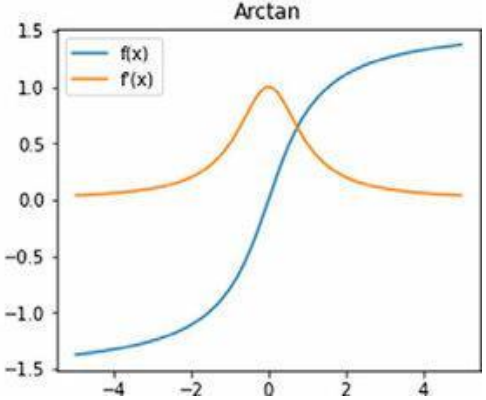
ReLU is usually the best default activation function - this is partially why











Link

Gradescope Participation Question

What algorithm do we use to compute the gradients of all the network weights?

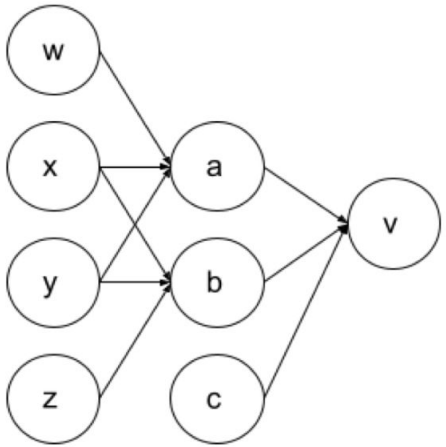
backprop

1. Neural Network Chain Rule Warm-Up

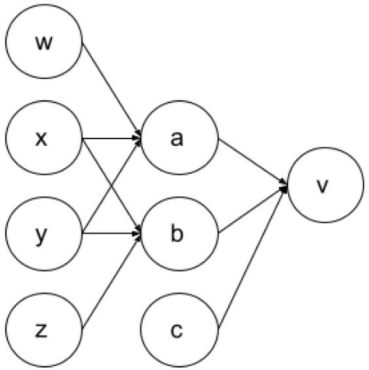
Consider the following equations:

$$v(a,b,c) = c(a-b)^2$$
$$a(w,x,y) = (w+x+y)^2$$
$$b(x,y,z) = (x-y-z)^2$$

The way variables are related to each other can be represented as the network:



(a) Using the multi-variate chain rule(part 1.b), write the derivatives of the output v with respect to each of the input variables: c, w, x, y, z using only partial derivative symbols.



(a) Using the multi-variate chain rule(part 1.b), write the derivatives of the output v with respect to each of the input variables: c, w, x, y, z using only partial derivative symbols.

$$
\begin{aligned}
\frac{\partial v}{\partial c} &= \frac{\partial v}{\partial c} \\
\frac{\partial v}{\partial w} &= \frac{\partial v}{\partial a} \cdot \frac{\partial a}{\partial w} \\
\frac{\partial v}{\partial x} &= \frac{\partial v}{\partial a} \cdot \frac{\partial a}{\partial x} + \frac{\partial v}{\partial b} \cdot \frac{\partial b}{\partial x} \\
\frac{\partial v}{\partial y} &= \frac{\partial v}{\partial a} \cdot \frac{\partial a}{\partial y} + \frac{\partial v}{\partial b} \cdot \frac{\partial b}{\partial y} \\
\frac{\partial v}{\partial z} &= \frac{\partial v}{\partial b} \cdot \frac{\partial b}{\partial z}
\end{aligned}
$$

(b) Compute the values of all the partial derivatives on the RHS of your results to the previous question. Then use them to compute the values on the LHS.

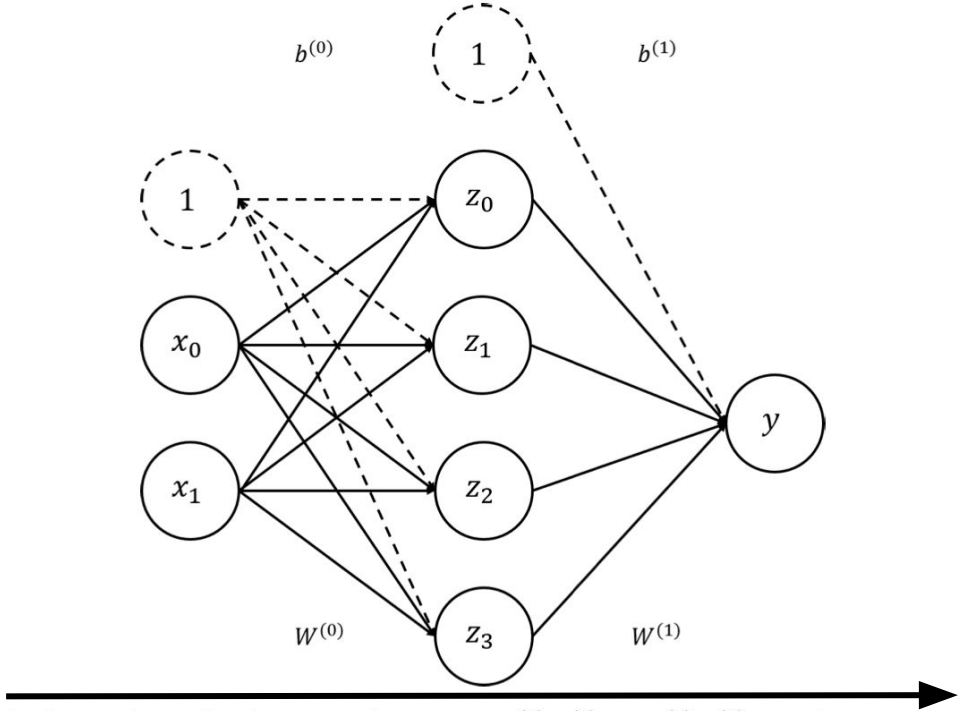
$$\frac{\partial v}{\partial a} = 2c(a-b) \quad \frac{\partial v}{\partial b} = -2c(a-b) \quad \frac{\partial v}{\partial c} = (a-b)^2$$
$$\frac{\partial a}{\partial w} = 2(w+x+y) \quad \frac{\partial a}{\partial x} = 2(w+x+y) \quad \frac{\partial a}{\partial y} = 2(w+x+y)$$
$$\frac{\partial b}{\partial x} = 2(x-y-z) \quad \frac{\partial b}{\partial y} = -2(x-y-z) \quad \frac{\partial b}{\partial z} = -2(x-y-z)$$

$$\frac{\partial v}{\partial c} = (a-b)^2$$
$$\frac{\partial v}{\partial w} = \frac{\partial v}{\partial a} \cdot \frac{\partial a}{\partial w} = 4c(a-b)(w+x+y)$$
$$\frac{\partial v}{\partial x} = \frac{\partial v}{\partial a} \cdot \frac{\partial a}{\partial x} + \frac{\partial v}{\partial b} \cdot \frac{\partial b}{\partial x} = 4c(a-b)(w+x+y) - 4c(a-b)(x-y-z) = 4c(a-b)(w+2y+z)$$
$$\frac{\partial v}{\partial y} = \frac{\partial v}{\partial a} \cdot \frac{\partial a}{\partial y} + \frac{\partial v}{\partial b} \cdot \frac{\partial b}{\partial y} = 4c(a-b)(w+x+y) + 4c(a-b)(x-y-z) = 4c(a-b)(w+2x-z)$$
$$\frac{\partial v}{\partial z} = \frac{\partial v}{\partial b} \cdot \frac{\partial b}{\partial z} = 4c(a-b)(x-y-z)$$

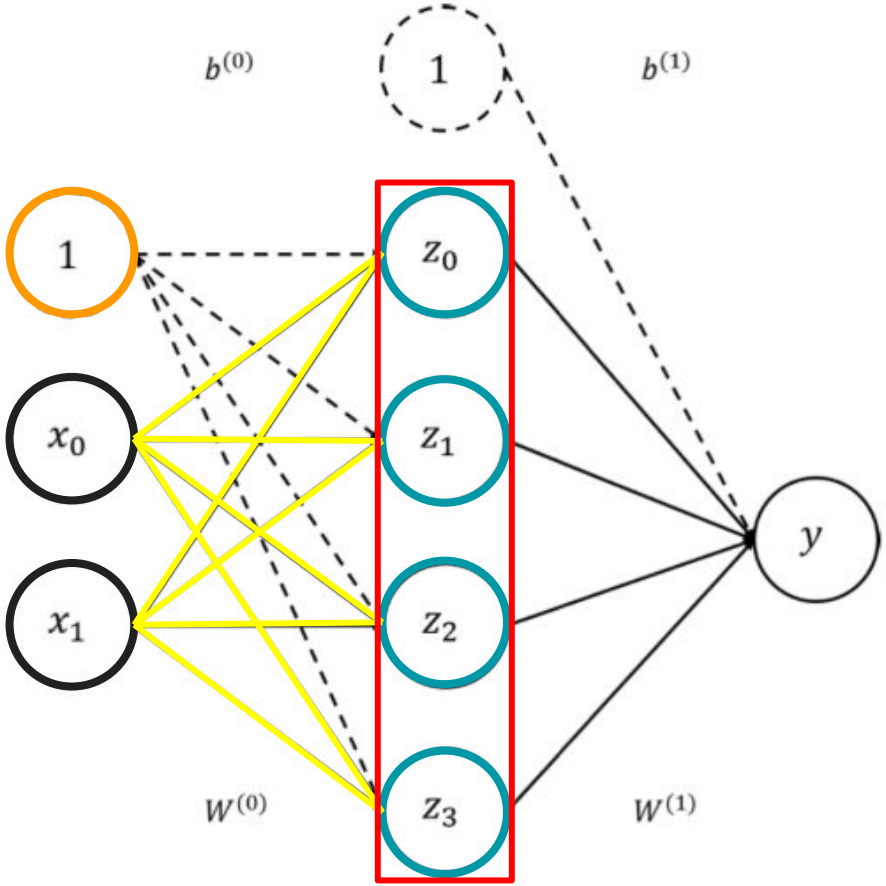
2. 1-Hidden-Layer Neural Network Gradients and Initialization

2.1. Forward and Backward pass

Consider a 1-hidden-layer neural network with a single output unit. Formally the network can be defined by the parameters $W^{(0)} \in \mathbb{R}^{h \times d}$, $b^{(0)} \in \mathbb{R}^h$; $W^{(1)} \in \mathbb{R}^{1 \times h}$ and $b^{(1)} \in \mathbb{R}$. The input is given by $x \in \mathbb{R}^d$. We will use sigmoid activation for the first hidden layer z and no activation for the output y. Below is a visualization of such a neural network with $d = 2$ and $h = 4$.

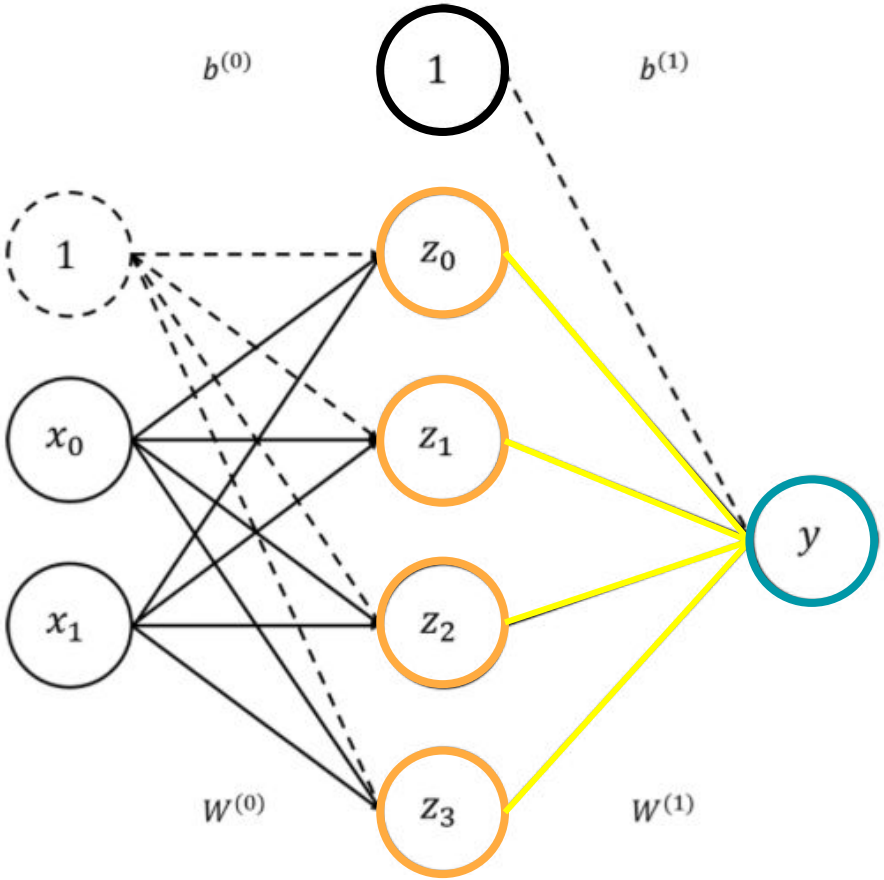


(a) Write out the forward pass for the network using $x, W^{(0)}, b^{(0)}, z, W^{(1)}, b^{(1)}, \sigma$ and y.

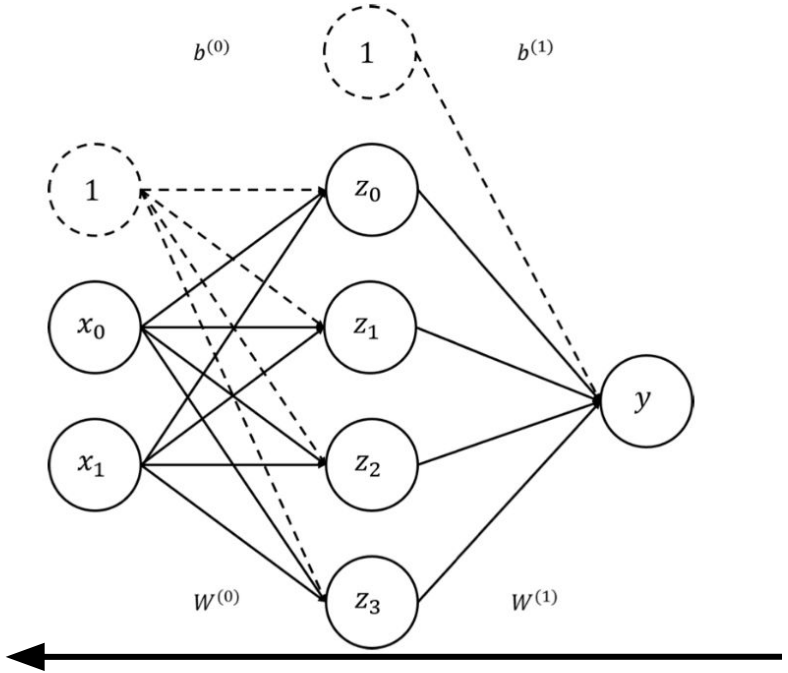


Apply sigmoid activation on z

$$z = \sigma(W^{(0)}x + b^{(0)})$$



$$y = W^{(1)}z + b^{(1)}$$



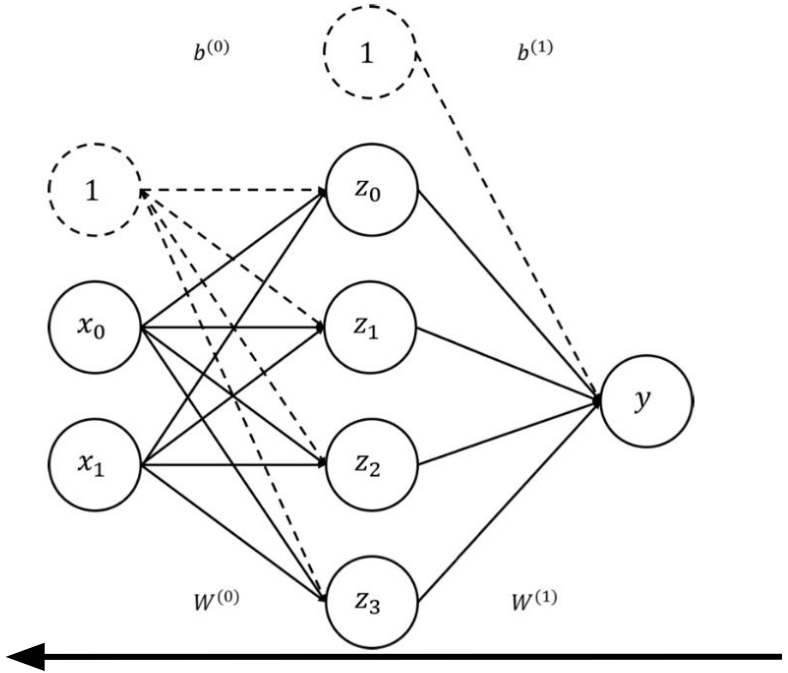
$$z = \sigma(W^{(0)}x + b^{(0)})$$

$$y = W^{(1)}z + b^{(1)}$$

Scalar derivative			Vector derivative		
$f(x)$	$\rightarrow$	$\frac{df}{dx}$	$f(\mathbf{x})$	$\rightarrow$	$\frac{df}{d\mathbf{x}}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{B}$	$\rightarrow$	$\mathbf{B}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{b}$	$\rightarrow$	$\mathbf{b}$
x^2	$\rightarrow$	$2x$	$\mathbf{x}^T\mathbf{x}$	$\rightarrow$	$2\mathbf{x}$
bx^2	$\rightarrow$	$2bx$	$\mathbf{x}^T\mathbf{B}\mathbf{x}$	$\rightarrow$	$2\mathbf{B}\mathbf{x}$

(b) Find the partial derivatives of the output with respect $W^{(1)}$ and $b^{(1)}$, namely $\frac{\partial y}{\partial W^{(1)}}$ and $\frac{\partial y}{\partial b^{(1)}}$.

$$\frac{\partial y}{\partial W^{(1)}} = z \qquad \frac{\partial y}{\partial b^{(1)}} = 1$$



$$z = \sigma(W^{(0)}x + b^{(0)})$$

$$y = W^{(1)}z + b^{(1)}$$

Scalar derivative			Vector derivative		
$f(x)$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}x}$	$f(\mathbf{x})$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{B}$	$\rightarrow$	$\mathbf{B}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{b}$	$\rightarrow$	$\mathbf{b}$
x^2	$\rightarrow$	$2x$	$\mathbf{x}^T\mathbf{x}$	$\rightarrow$	$2\mathbf{x}$
bx^2	$\rightarrow$	$2bx$	$\mathbf{x}^T\mathbf{B}\mathbf{x}$	$\rightarrow$	$2\mathbf{B}\mathbf{x}$

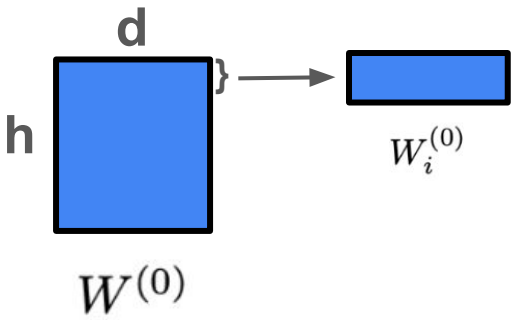
(c) Now find the partial derivative of the output with respect to the output of the hidden layer z, that is $\frac{\partial y}{\partial z}$

$$\frac{\partial y}{\partial z} = W^{(1)}$$

(d) Finally find the partial derivatives of the output with respect to $W^{(0)}$ and $b^{(0)}$, that is $\frac{\partial y}{\partial W^{(0)}}$ and $\frac{\partial y}{\partial b^{(0)}}$.

$z = \sigma(W^{(0)}x + b^{(0)})$ $\quad y = W^{(1)}z + b^{(1)}$

$\sigma'(a) = \sigma(a) * (1 - \sigma(a))$



$z_i = \sigma(W_i^{(0)}x + b_i^{(0)})$ $\quad \mathbb{R}^{1\times d}$ $\quad \mathbb{R}^{d\times 1}$

$$\frac{\partial z_i}{\partial W_i^{(0)}} = z_i(1-z_i)x^\top$$

$y = W^{(1)}z + b^{(1)}$

$$\frac{\partial y}{\partial z_i} = W_i^{(1)} \qquad W^{(1)} \in \mathbb{R}^{1\times h}$$

$$\frac{\partial y}{\partial W_i^{(0)}} = \frac{\partial y}{\partial z_i}\frac{\partial z_i}{\partial W_i^{(0)}} = W_i^{(1)} \cdot z_i(1-z_i)x^\top \in \mathbb{R}^{1\times d}$$

Generalizing for all h rows…

$$\frac{\partial y}{\partial W^{(0)}} = \left[W^{(1)^\top} \circ z \circ (1-z)\right] x^\top \in \mathbb{R}^{h\times d}$$

(d) Finally find the partial derivatives of the output with respect to $W^{(0)}$ and $b^{(0)}$, that is $\frac{\partial y}{\partial W^{(0)}}$ and $\frac{\partial y}{\partial b^{(0)}}$.

$z = \sigma(W^{(0)}x + b^{(0)})$ $\quad y = W^{(1)}z + b^{(1)}$

$\sigma'(a) = \sigma(a) * (1 - \sigma(a))$

Repeat the same steps for bias…

$$z_i = \sigma(W_i^{(0)}x + b_i^{(0)}) \longrightarrow \frac{\partial z_i}{\partial b_i^{(0)}} = z_i(1 - z_i)$$

$$y = W^{(1)}z + b^{(1)} \longrightarrow \frac{\partial y}{\partial z_i} = W_i^{(1)} \qquad W^{(1)} \in \mathbb{R}^{1 \times h}$$

$$\frac{\partial y}{\partial b_i^{(0)}} = \frac{\partial y}{\partial z_i}\frac{\partial z_i}{\partial b_i^{(0)}} = W_i^{(1)} \cdot z_i(1 - z_i) \in \mathbb{R}$$

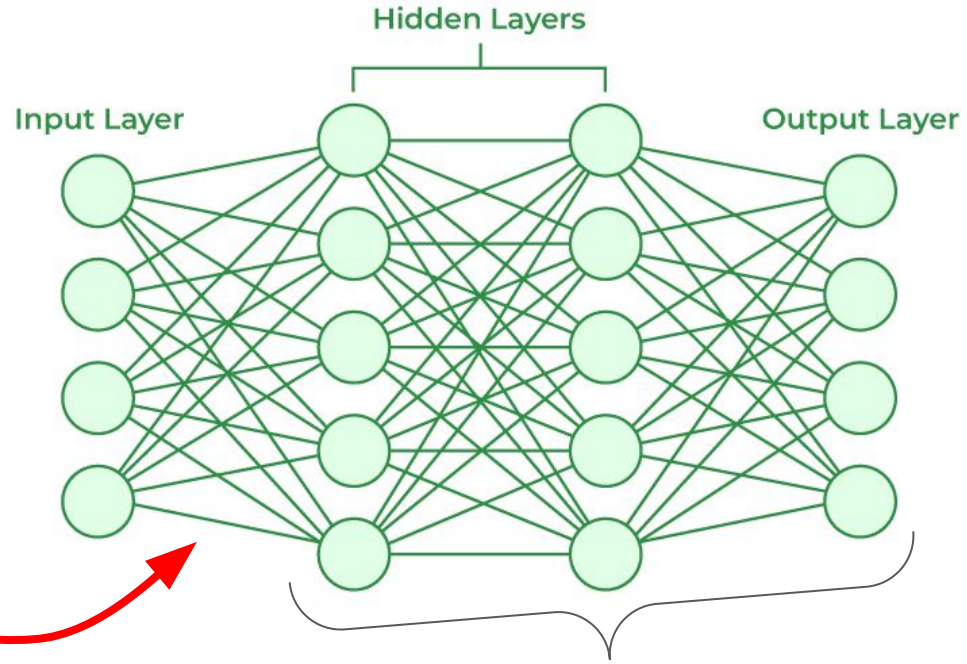
↓ Generalizing for all h rows…

$$\frac{\partial y}{\partial b^{(0)}} = W^{(1)^\top} \circ z \circ (1 - z) \in \mathbb{R}^h$$

Initializations

Initializations are incredibly important

What happens if we set all weights to 0 initially in a neural network?



No matter the input, all outputs will be the same…

Everything stays the same here too, no learning!


AY PD

Interactive Initialization Example

2.2. Weight initialization

Suppose we initialize all weights and biases in the network to 0 before performing gradient descent.

(a) For all $x \in \mathbb{R}^d$, find z and y after the forward pass.

$$z_i = \sigma(W_i^{(0)} x + b^{(0)_i}) = \sigma(\mathbf{0}x + 0) = \sigma(0) = \frac{1}{2}$$

$$y = W^{(1)} z + b^{(1)} = \mathbf{0} \cdot \frac{\mathbf{1}}{\mathbf{2}} + 0 = 0$$

(b) Now find the values of the gradients $\frac{\partial y}{\partial W^{(1)}}$, $\frac{\partial y}{\partial b^{(1)}}$, $\frac{\partial y}{\partial W^{(0)}}$ and $\frac{\partial y}{\partial b^{(0)}}$. Note that some of the gradients will be in terms of x.

$$
\begin{aligned}
\frac{\partial y}{\partial W^{(1)}} &= z = \mathbf{\frac{1}{2}} \\
\frac{\partial y}{\partial b^{(1)}} &= 1 \\
\frac{\partial y}{\partial W^{(0)}} &= \left[W^{(1)} \circ z \circ (1-z)\right] x^\top \\
&= (\mathbf{0} \circ \mathbf{\frac{1}{2}} \circ \mathbf{\frac{1}{2}}) x^\top = \mathbf{0} \\
\frac{\partial y}{\partial b^{(0)}} &= W^{(1)} \circ z \circ (1-z) \\
&= \mathbf{0} \circ \mathbf{\frac{1}{2}} \circ \mathbf{\frac{1}{2}} = \mathbf{0}\,.
\end{aligned}
$$

(c) Observe the values of each z_i and observe each $\frac{\partial y}{\partial W_i^{(l)}}$ and $\frac{\partial y}{\partial b_i^{(l)}}$. What do you notice? And what does this imply for the expressiveness of the network? (Note that there is nothing special about the value 0 here, it just simplifies the calculations. The same can be shown for initialization with any constant c)

The key insight is that if we initialize the weights to all have the same value, all z_i are the same. Similarly all $W_i^{(l)}$ and $b_i^{(l)}$ are the same too and so the output y could be expressed with just a single z_i instead of h. Thus the neural network boils down to just having a single hidden unit. The same holds for the gradients, so during a step of gradient descent, $W_i^{(l)}$ and $b_i^{(l)}$ are updated in the same way. Thus after a step of gradient descent, all $W_i^{(l)}$ and $b_i^{(l)}$ are still the same. By induction, the same holds after an arbitrary number of steps of gradient descent.

IF THIS DOES NOT MAKE SENSE JUST IGNORE IT, THIS IS NOT REQUIRED

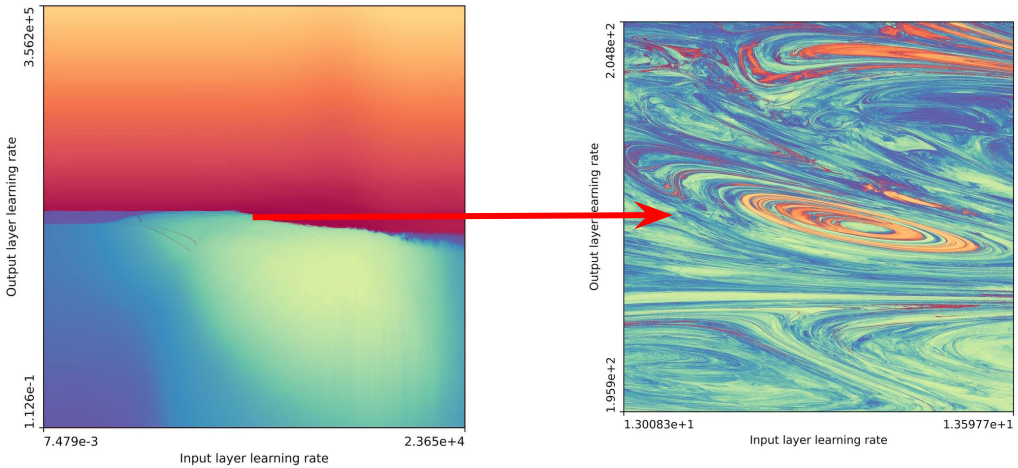
Sensitive dependence in initializations

input layer: 16 neurons,
hidden layer: 16 neurons,
output layer: 1 neuron, tanh activation

I'm not an expert in fractals or chaos theory, but thought this was super cool:

(video links in paper)

[The boundary of neural network trainability is fractal]



Blue-green = convergence
Red-yellow = failure

Questions/Chat Time!