Neural Networks from Scratch

Neural Networks from Scratch Pt 2

Caleb Hallinan

01/07/2026

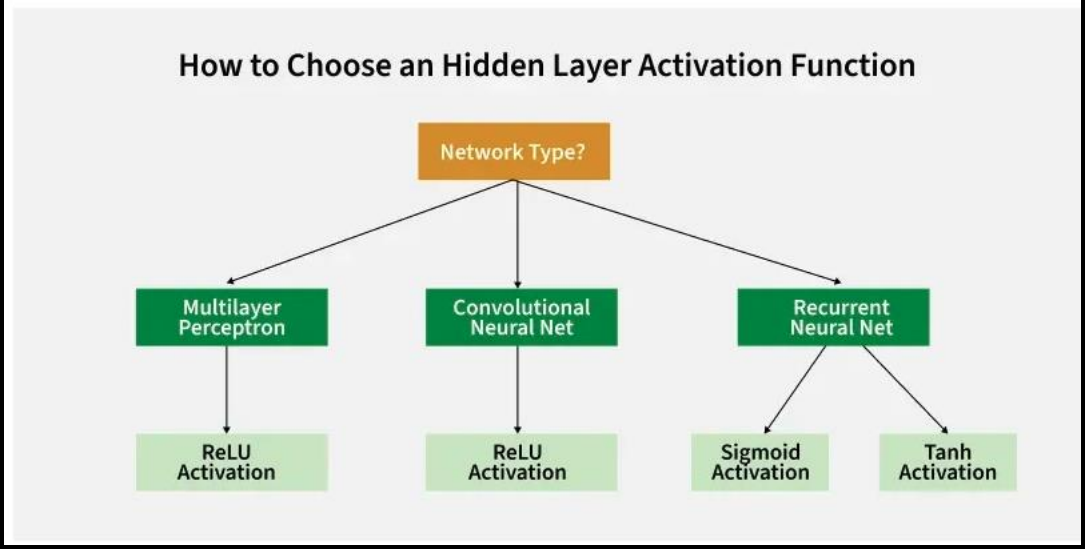
Announcements

- Lecture and code from yesterday is on GitHub
- Today's code also on GitHub if you want to follow along
  - Going to try and make CNN and PyTorch code more interactive!

A few questions from reflection cards

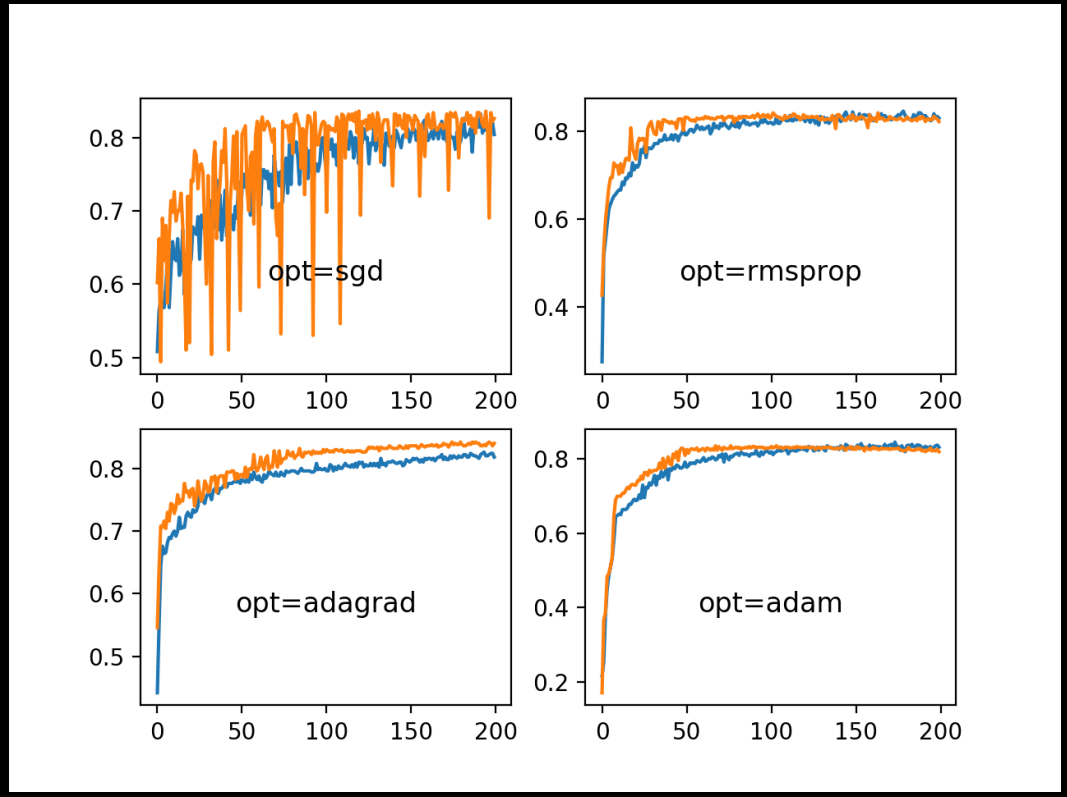
How do you choose the best activation function?

- There is no universal rule for choosing the best activation functions – comes down to trial and error
  - However, there are some ideas for when it is best to use each one
  - https://www.geeksforgeeks.org/deep-learning/choosing-the-right-activation-function-for-your-neural-network/
- General rules:
  - ReLU or ReLU variant in hidden layers
  - Sigmoid or Softmax for last layer (why?)



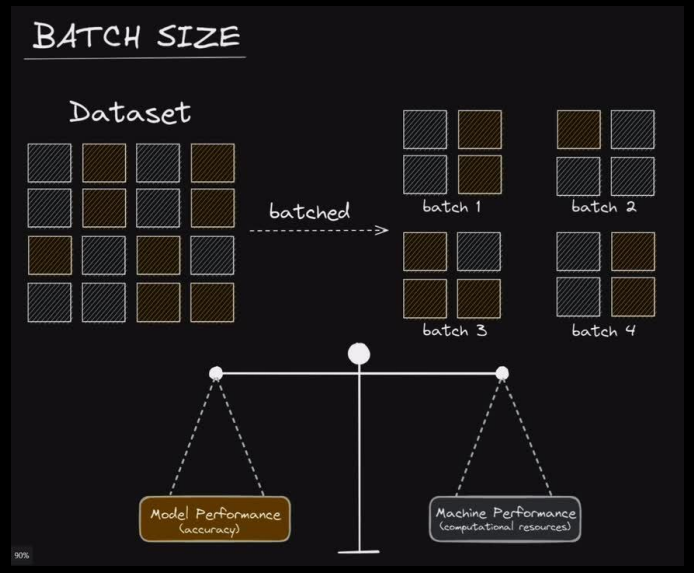
How do adaptive learning rates work?

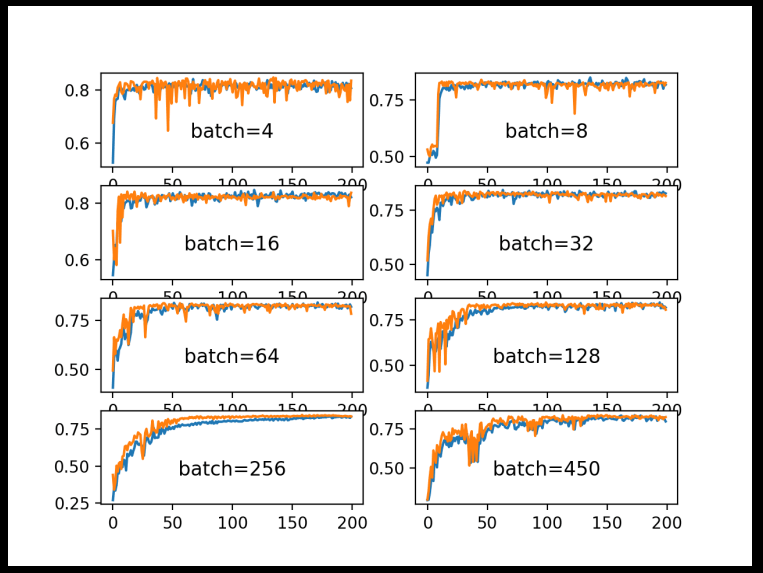
- We will go over this in more detail tomorrow!
  - Specifically, we will discuss learning rate decay and momentum



Why use batches?

- Computing gradients on a small set of samples at once is much faster and more memory-efficient than processing the entire dataset every step, especially on GPUs
- Give a “noisy” but useful gradient estimate that often helps optimization and generalization compared to purely full-dataset updates.





Why is np.dot(AxB) the same as np.dot(AxB^T)

Depends on how the weights are stored

```
# Dense layer
class Layer_Dense:
    # Layer Initialization
    def __init__(self, n_inputs, n_neurons):
        # Initialize weights and biases
        self.weights = 0.01 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))

    # Forward Pass
    def forward(self, inputs):
        # Calculate output values from inputs, weights and biases
        self.output = np.dot(inputs, self.weights) + self.biases
```

Our code from yesterday

If weights are initialized as (n_inputs, n_neurons) then it is already ready to be multiplied to the inputs because (batch, n_inputs) x (n_inputs, n_neurons) is good!

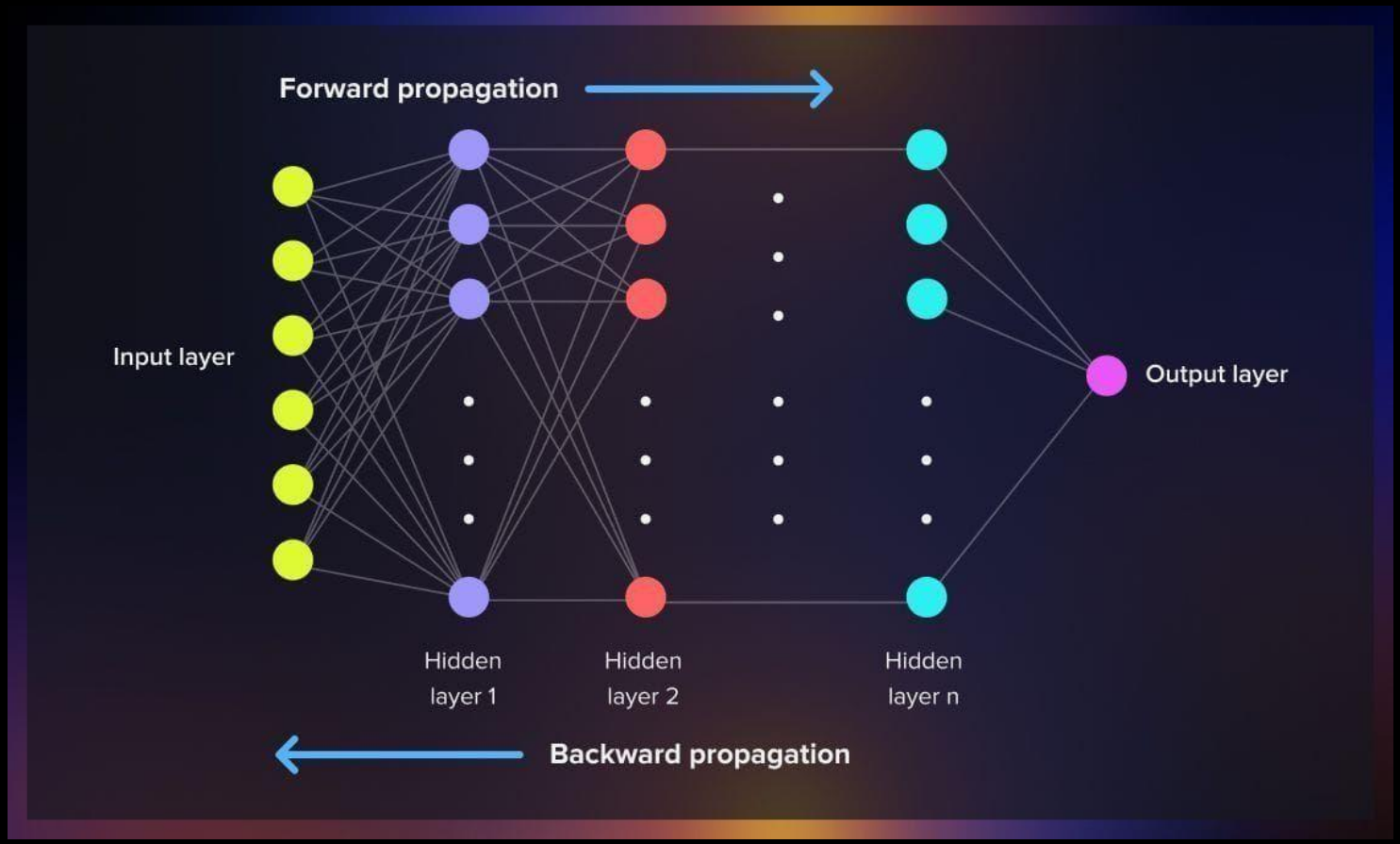
If it was instead (n_neurons, n_ inputs), then you would need to transpose them

Outline for today

- Continue going through neural networks from scratch – building from the ground up

Today’s notes and coding

Last Class: How do NN do forward propagation?



This class: How do NN do backward propagation?

Derivatives

- shows impact of w, b on loss
- need to know effect of x on y
or Δx on Δy

* the impact of a param on output

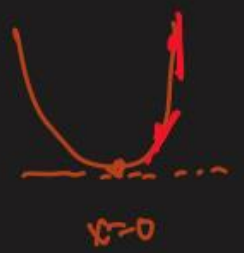
⇒ $f(x) = 2x$

$\rightarrow$ slope $= \frac{y_2 - y_1}{x_2 - x_1} = 2$

constant slope

⇒ $f(x) = 2x^2$

- we see as x increases y increases drastically
- we measure the slope of the tangent point



$x=0$ $\Delta x \cdot \Delta y$

$x=1$ $\Delta x \rightarrow \Delta y \uparrow$

$x=5$ $\Delta x \rightarrow \Delta y \uparrow\uparrow$

So, to measure the impact which x has on y, we measure the slope of the tangent line at x

= **Derivative!!**

If we change x, how does y change

Derivative notation: d/dx

So if $ax^n \rightarrow a \cdot n x^{n-1}$

ex. Ask students derivatives

$f(x) = 7$

$df/dx(x) = 1 = f'(x)$

$f(x) = x^2$

$f'(x) = 2x$

$f(x) = x^3 + 2x^2 + 6$

$df/dx = 3x^2 + 4x + 0$

Gradients, Partial Derivatives, chain rule

what about $f(x, y, z) \rightarrow f'(x, y, z)$?

PD notation: $f(x, y, z)$

- $\partial/\partial x \; f(x, y, z)$ PD of f wrt x
- $\partial/\partial y \; f(x, y, z)$ PD of f wrt y
- $\partial/\partial z \; f(x, y, z)$

ex. $f(x,y) = 2x + 3y^2$

Does anyone want to show how to solve this?

$\partial f/\partial x = \partial/\partial x (2x + 3y^2)$

$= \partial/\partial x (2x) + \partial/\partial x (3y^2)$

$= 2 + 0$

$= 2$

$\partial f/\partial y = \partial/\partial y (2x) + \partial/\partial y (3y^2)$

$= 6y$

- treat all variables that aren't being derived as constants
- these are calc slopes in specific directions
- $\partial/\partial x$ finds slope in x direction

ex.

$f(x,y,z) = 3x^3z - y^2 + 5z + 2yz$

so $\partial/\partial x = \partial/\partial x (3x^3z) - \partial/\partial x (y^2) + \partial/\partial x (5z) + \partial/\partial x (2yz)$

(0) (0) (0)

Bring out constants

$= 3z(3x^2)$

$= 9x^2z$

ex.

$f(x,y) = \max(x,y) \Rightarrow x \begin{cases} x \geq y \\ x < y \end{cases}$

ReLU?!

$f(x,y) = \max(0,x)$

$\partial f/\partial x = \partial/\partial x (\max(x,y))$

$= 1$ if $x > y$

$= 0$ if $x < y$

$\partial f/\partial x = \partial/\partial x (\max(0,x))$

$= 1$ if $x > 0$

$= 0$ if $x < 0$

- need this b/c NN have tons of parameters → need to know how to find their effects on the function

Gradient Descent

Say we have our same example

$$f(x,y,z) = 3x^3z - y^2 + 5z + 2yz$$

$$\partial f/\partial x = 9x^2z, \quad \partial f/\partial y = -2y + 2z, \quad \partial f/\partial z = 3x^3 + 5 + 2y$$

Gradient is represented as a vector like:

$$\nabla f(x,y,z) = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \\ \partial f/\partial z \end{bmatrix} = \begin{bmatrix} 9x^2z \\ -2y + 2z \\ 3x^3 + 5 + 2y \end{bmatrix}$$

3×1 $\qquad$ 3×1

Contains direction of the steepest descent

→ we want to find the direction where the loss decreases the most

Chain rule = backbone of backprop

When you have functions built from other functions, you can use the chain rule to take the derivative by: differentiating the outside function (keeping the inside the same), then multiply by the derivative of the inside

Does anyone know the formula for the chain rule?

$$dy/dx = f'(g(x)) \cdot g'(x)$$

where $y = f(g(x))$

To solve:

let $u = g(x)$, $y = f(u)$

so $dy/dx = dy/du \cdot du/dx$

Ex. $y = (3x^2 - 5x + 1)^7$

what would u be?

$u = 3x^2 - 5x + 1$ s.t.

$y = u^7$

so $dy/du = 7u^6$ and

$du/dx = 6x - 5$ so

$dy/dx = 7u^6 (6x - 5)$ ✓

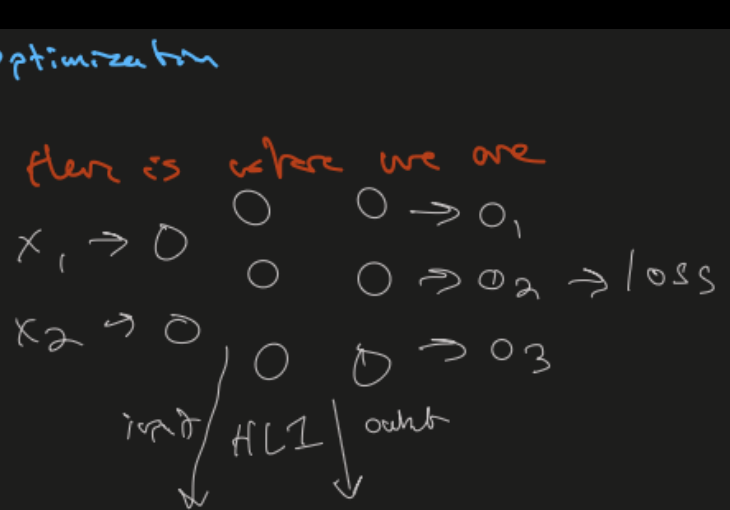
$dy/dx = 7(3x^2 - 5x + 1)^6 (6x - 5)$

A NN is a chain of chain rules
Would look like

F(g(h(z(x))))

Optimization

Here is where we are



We know how to calc loss, but how do we use it to learn?

Say we have this landscape

How do you think we can adjust weights, biases to decrease loss?

This is optimization!

Strategy 1

① pick random W, b
② find loss
③ iterate 100000 times
④ find W, b w/ lowest loss

Ex on iexpub

Strategy 2

① init random W, b
② randomly adjust W, b
③ if loss decreases, then adjust next W, b to be close to old W, b
- if loss increases, don't update weights, biases
④ keep track of lowest loss

Best Method

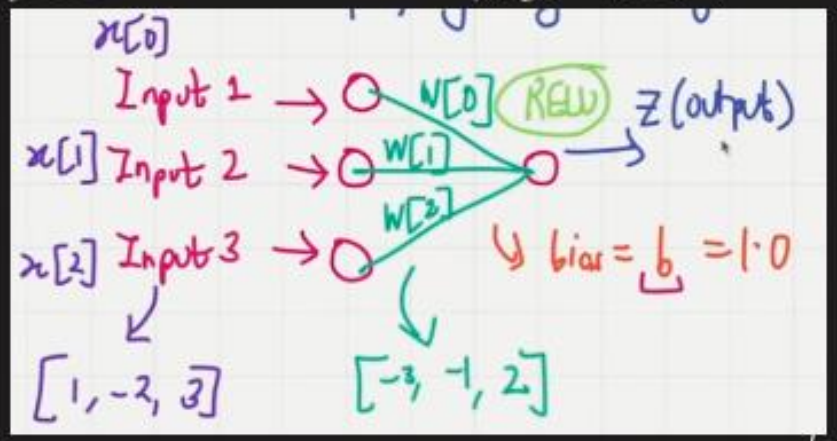
Go along negative gradient of slope

negative gradient shows the steepest descent

Monday, March 24, 2025 11:51 AM

Backpropagation

single neuron



$z = ReLU(x[0]*w[0] + \dots)$

$z = max(xw_0 + \dots + b, 0)$

$Loss = (z-0)^2 = z^2$

↳ to minimize, use GD

So how do we optimize the weights & biases of neurons so that our output is close to 0?

** Let us minimize the loss function **

→ We need to find how to update w[0], w[1], w[2], b so that loss is minimized.

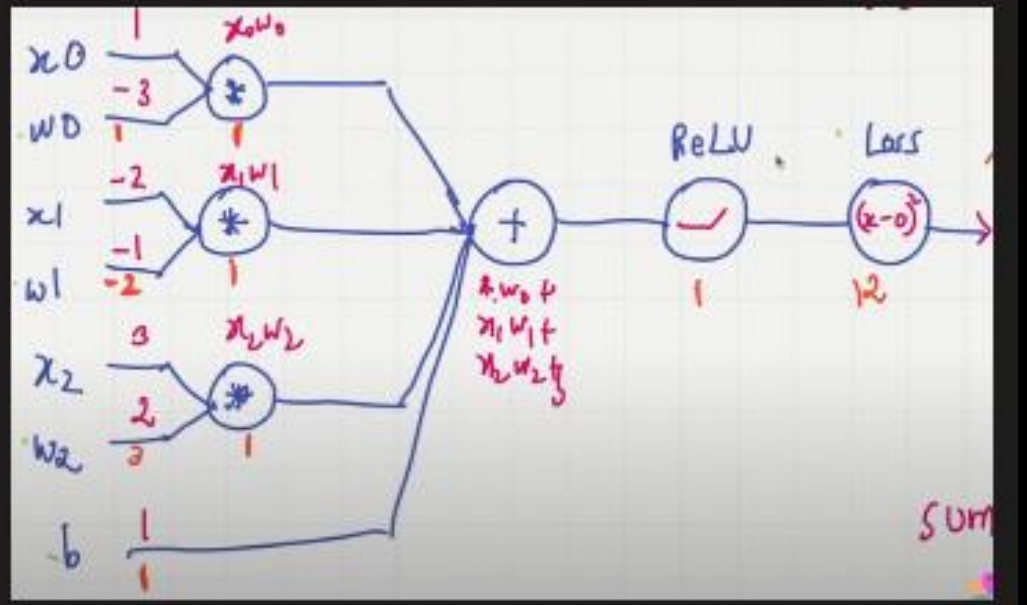
→ For this, we have to move in negative gradient direction

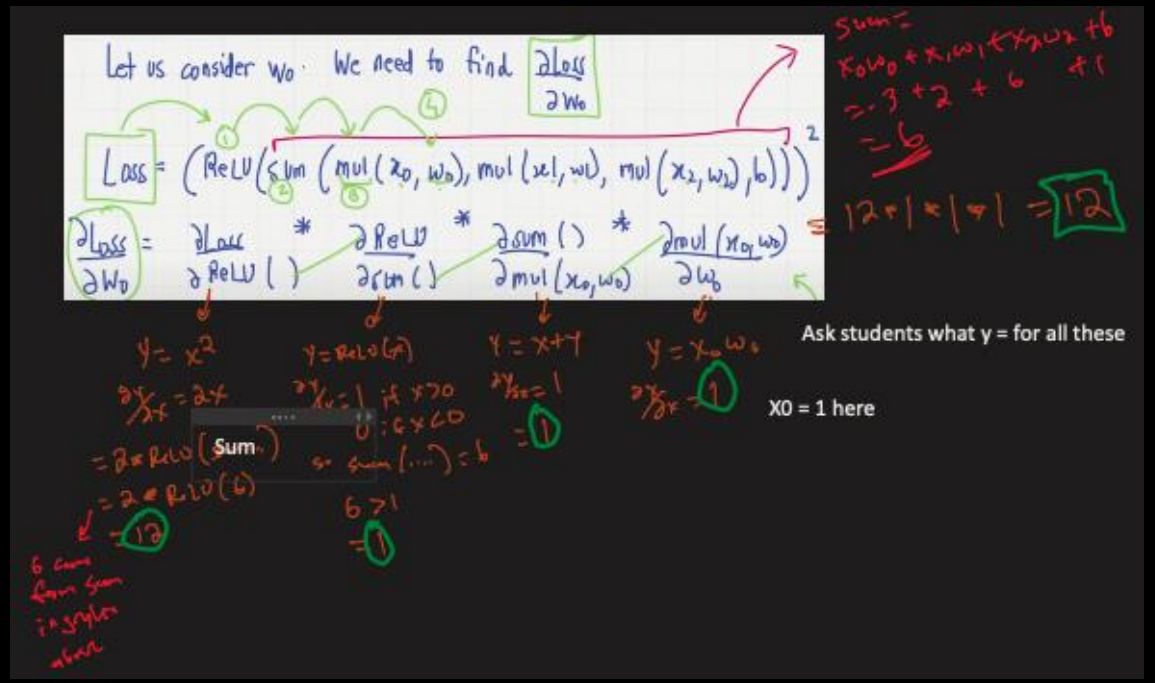
We have to find derivative of loss with respect to: w[0], w[1], w[2], b

Want to find what changes these

$\left(\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial b}\right)$ → so we change w_0 by

$w_0 \to w_0 - \eta \frac{\partial L}{\partial w_0}$ (learning constant)





$$\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial ReLU()} * \frac{\partial ReLU}{\partial sum()} * \frac{\partial sum()}{\partial mul(x_1, w_1)} * \frac{\partial mul(x_1, w_1)}{\partial w_1} = 12 * 1 * 1 * -2 = -24$$

$x_1 = -2$

$$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial ReLU()} * \frac{\partial ReLU}{\partial sum()} * \frac{\partial sum()}{\partial mul(x_2, w_2)} * \frac{\partial mul(x_2, w_2)}{\partial w_2} = 12 * 1 * 1 * 3 = 36$$

$x_2 = 3$

$$\frac{\partial Loss}{\partial b} = \frac{\partial Loss}{\partial ReLU()} * \frac{\partial ReLU}{\partial sum()} * \frac{\partial sum()}{\partial b} = 12 * 1 * 1 = 12$$

Gradient descent:

$w_0 = w_0 - 0.001 * \frac{\partial Loss}{\partial w_0}$

$= -3 - 0.001 * (12) = -3.012$

$w_1 = w_1 - 0.001 * \frac{\partial Loss}{\partial w_1}$

$= -1 - 0.001 * (-24) = -0.976$

$w_2 = w_2 - 0.001 * \frac{\partial Loss}{\partial w_2}$

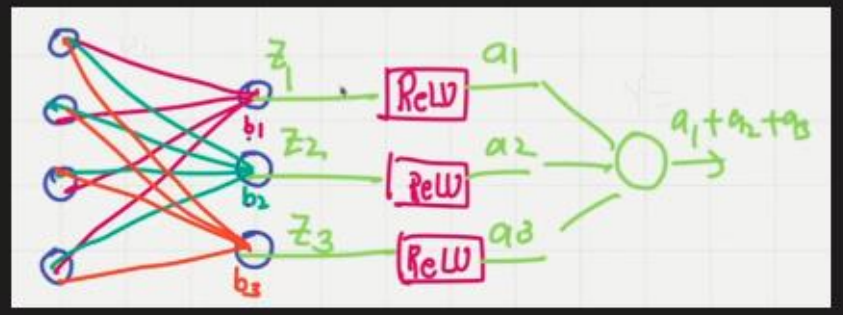
Earlier loss = 36

New loss = $[ReLU(sum(..))]^2$

$= (-3.012) - 2(-0.976) + 3(1.964) + 0.988$

$= (5.82)$

$Loss_{new} = (5.82)^2 = 33.8724$



Want loss to be as close to 0 as possib

$Loss = (a_1 + a_2 + a_3)^2$

$= y^2$

Y = (a1 + a2 + a3)
So that's why deriv of y wrt a1 = 1

Notation:

$z_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1$ $\quad a_1 = ReLU(z_1)$

$z_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2$ $\quad a_2 = ReLU(z_2)$

$z_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3$ $\quad a_3 = ReLU(z_3)$

How would you write this?

Backward pass: $\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial a_1} * \frac{\partial a_1}{\partial z_1} * \frac{\partial z_1}{\partial w_{11}} = 2y * 1_{(z_1>0)} * x_1$

$2y$ $\quad 1$ $\quad 1_{(z_1>0)}$ $\quad x_1$

diff is x

et.2 $\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial a_1} * \frac{\partial a_1}{\partial z_1} * \frac{\partial z_1}{\partial w_{12}} = 2y * 1_{(z_1>0)} * x_2$

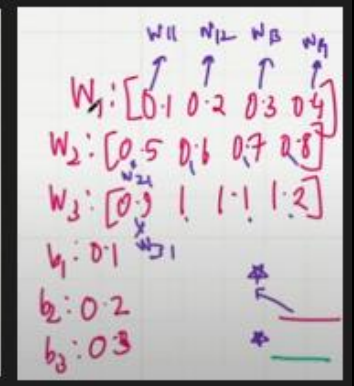
$2y$ $\quad 1$ $\quad 1_{(z_1>0)}$ $\quad x_2$

Forward pass: $z_1 = 3$; $z_2 = 7.2$; $z_3 = 11.4$

$a_1 = 3$; $a_2 = 7.2$; $a_3 = 11.4$

$y = a_1 + a_2 + a_3 = 21.6$

$L = y^2 = 466.56$



Inputs = [1,2,3,4]

Did all these partial derivates to finally update weights

$\frac{\partial L}{\partial w_{11}} = 2y * x_1 = 43.2 * 1 = 43.2$; $\frac{\partial L}{\partial w_{12}} = 2y * x_2 = 86.4$; $\frac{\partial L}{\partial w_{13}} = 129.6$; $\frac{\partial L}{\partial w_{14}} = 172.8$

$\frac{\partial L}{\partial w_{21}} = 2y * x_1 = 43.2$; $\frac{\partial L}{\partial w_{22}} = 2y * x_2 = 86.4$; $\frac{\partial L}{\partial w_{23}} = 129.6$; $\frac{\partial L}{\partial w_{24}} = 172.8$

$\frac{\partial L}{\partial w_{31}} = 2y * x_1 = 43.2$; $\frac{\partial L}{\partial w_{32}} = 86.4$; $\frac{\partial L}{\partial w_{33}} = 129.6$; $\frac{\partial L}{\partial w_{34}} = 172.8$

$\frac{\partial L}{\partial b_1} = 2y * 1 = 43.2$; $\frac{\partial L}{\partial b_2} = 2y * 1 = 43.2$; $\frac{\partial L}{\partial b_3} = 2y * 1 = 43.2$

Example with W11

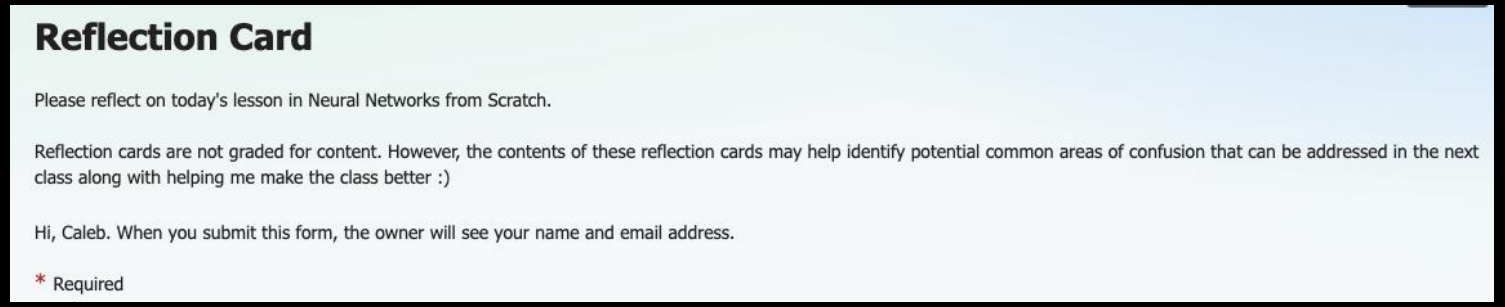
Gradient Descent update:

$W_{11}^{new} = W_{11}^{old} - 0.001^{stepsize} * \frac{\partial L}{\partial w_{11}} = 0.1 - 0.001 * 43.2 = 0.0568$

→ Do this w/ all weights

Now can check new loss value based on new weights!

Reflection Cards



Reflection Card

Please reflect on today's lesson in Neural Networks from Scratch.

Reflection cards are not graded for content. However, the contents of these reflection cards may help identify potential common areas of confusion that can be addressed in the next class along with helping me make the class better :)

Hi, Caleb. When you submit this form, the owner will see your name and email address.

* Required

- Essentially a means to help me make this class better!
  - Would you be interested in a hybrid version of class?

1. What is something that you learned in today's lecture?
2. What is something that you are still confused about from today's lecture?
3. Do you have any other comments/feedback/thoughts/suggestions/concerns?

https://forms.office.com/r/Kv8LtW4iJH