Neural Networks from Scratch

Neural Networks from Scratch Pt 3

Caleb Hallinan

01/08/2026

10x Xenium Breast Cancer Data

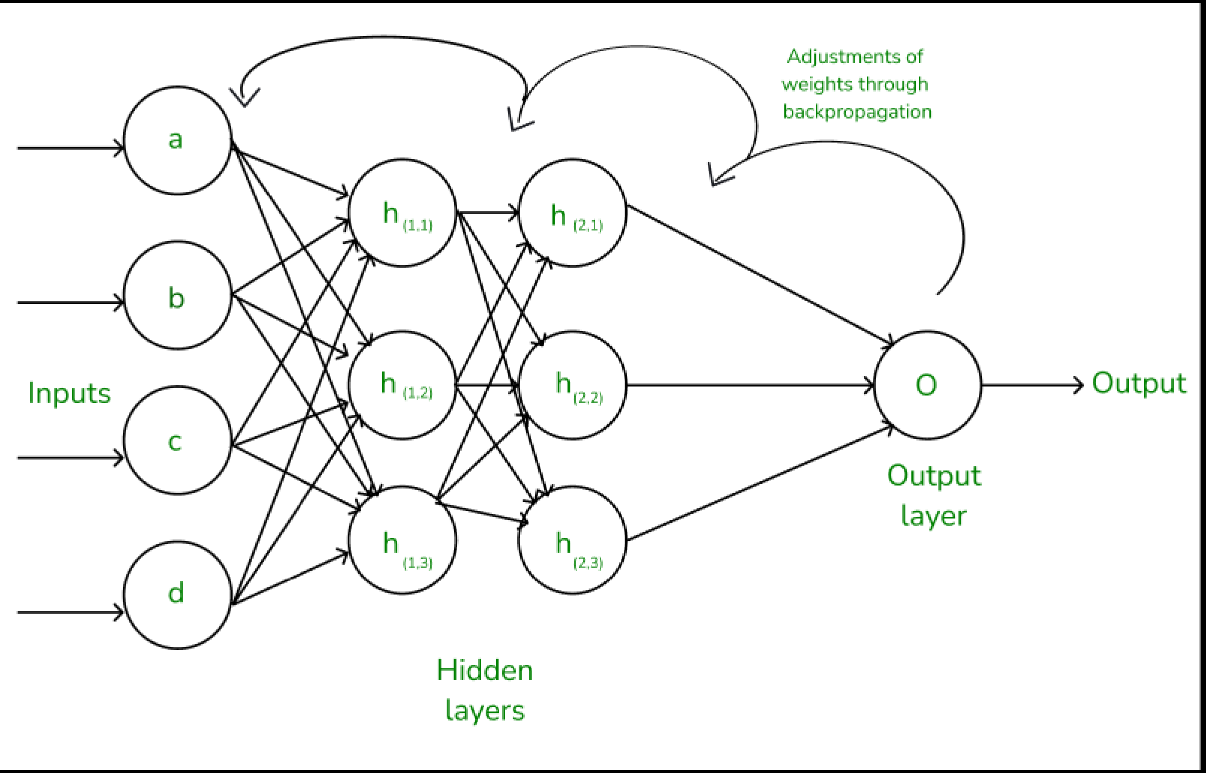
Announcements

- Lecture and code from yesterday is on GitHub
- Today’s code also on GitHub and there are exercises to complete while going through the code!
- I will try and find a room to do a hybrid class beginning next week

A few questions from reflection cards

How would the gradient computation look for different hidden layers?

- We will go over that first thing this morning!



Application of gradient descent and how we choose the value by which we multiply the derivatives by to establish our new loss

- Gradient descent updates weights by **stepping opposite the gradient**, scaled by a chosen **learning rate**, and the loss changes only as a consequence of that weight update.

Updating weights with no learning rate would cause too large or too small of steps

$$w = w - \frac{\partial L}{\partial w}$$

Scaling the step allows us to control how much we step

$$\Delta w = -\eta \frac{\partial L}{\partial w}$$

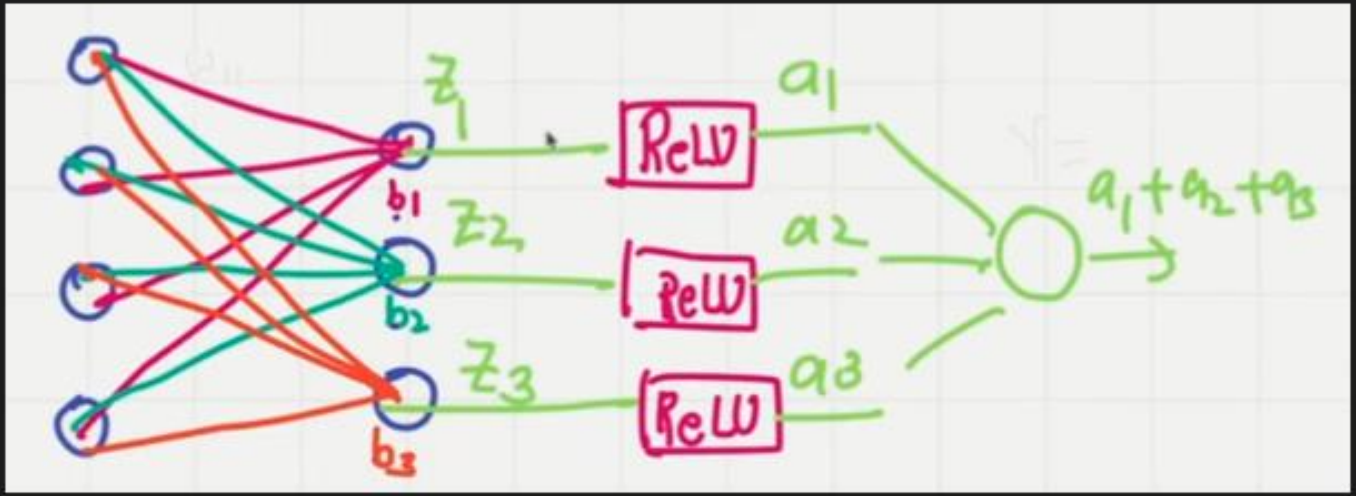
Chosen by experience, trial and error, or learning rate schedules or optimizers (ex. decay/momentum)

Typical values:

- $10^{-1} = 0.1$
- $10^{-2} = 0.01$
- $10^{-3} = 0.001$

Outline for today

- Finish neural networks from scratch – decay/momentum and multiple neurons
- Intro to PyTorch



Notation:

$z_1 = W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + W_{14}x_4 + b_1$

$z_2 = W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + W_{24}x_4 + b_2$

$z_3 = W_{31}x_1 + W_{32}x_2 + W_{33}x_3 + W_{34}x_4 + b_3$

$a_1 = ReLU(z_1)$

$a_2 = ReLU(z_2)$

$a_3 = ReLU(z_3)$

Want loss to be as close to 0 as possible

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

$= y^2$

Y = (a1 + a2 + a3)

So that's why deriv of y wrt a1 = 1

How would you write this?

Backward pass:

$$\frac{\partial L}{\partial W_{11}} = \underbrace{\frac{\partial L}{\partial y}}_{2y} * \underbrace{\frac{\partial y}{\partial a_1}}_{1} * \underbrace{\frac{\partial a_1}{\partial z_1}}_{1_{(z_1>0)}} * \underbrace{\frac{\partial z_1}{\partial W_{11}}}_{x_1} = 2y * 1_{(z_1>0)} * x_1$$

diff is x

eg. 2

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

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

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

$y = a_1 + a_2 + a_3 = 21.6$

$L = y^2 = 466.56$

$$\begin{array}{ccccc} & w_{11} & w_{12} & w_{13} & w_{14} \\ W_1: & 0.1 & 0.2 & 0.3 & 0.4 \end{array}$$

$W_2: [0.5\ \ 0.6\ \ 0.7\ \ 0.8]$ (w_{21})

$W_3: [0.9\ \ 1\ \ 1.1\ \ 1.2]$ (w_{31})

$b_1: 0.1$

$b_2: 0.2$

$b_3: 0.3$

Inputs = [1,2,3,4]

Did all these partial derivates to finally update weights

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

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

$$\frac{\partial L}{\partial W_{31}} = \overset{2y^* x_2}{\underline{43.2}};\ \frac{\partial L}{\partial W_{32}} = 86.4,\ \frac{\partial L}{\partial W_{33}} = 129.6\ \ \frac{\partial L}{\partial W_{34}} = 172.8$$

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

Example with W11

Gradient Descent update:

$$\underset{\substack{\downarrow \\ \text{new}}}{W_{11}} = \underset{\substack{\downarrow \\ \text{old}}}{W_{11}} - \underbrace{0.001}_{\text{step size}} * \frac{\partial L}{\partial W_{11}} = 0.1 - 0.001 * 43.2 = 0.0568$$

$\rightarrow$ Do this w/ all Weights

Learning rate decay

$$\alpha = \frac{\alpha_0}{1 + decay * t}$$

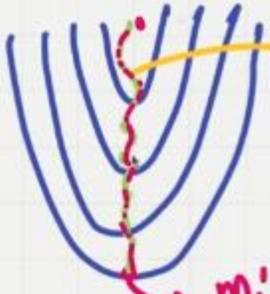
α_0 → initial stepsize

t → iteration number

$decay$ → learning rate decay

Reduces oscillations and helps to converge

Gradient Descent with Momentum



How it works? ⇒ PAST MATTERS

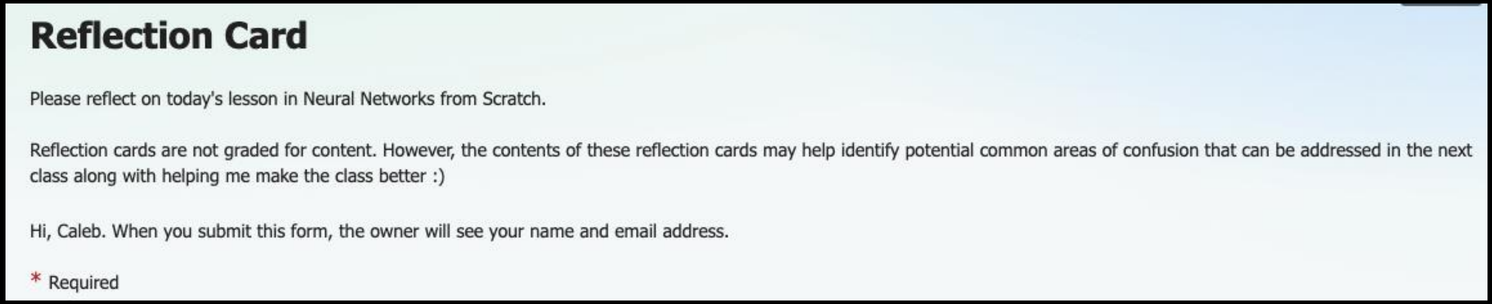
↳ Momentum uses previous update direction to influence next update

$$\boxed{\text{Weight update}} = \text{Momentum factor} * \text{previous weight updates} - \alpha \frac{\partial L}{\partial W}$$

$$\text{Weight}_{new} = \text{weight}_{old} + \text{Weight update}$$

$$W^{(t-1)} - \alpha \underbrace{\frac{\partial L}{\partial W}}_{\text{current gradient}} + \underbrace{\text{Mom. factor} * \text{Previous weight updates}}_{\text{Direction of previous changes}}$$

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!

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