Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

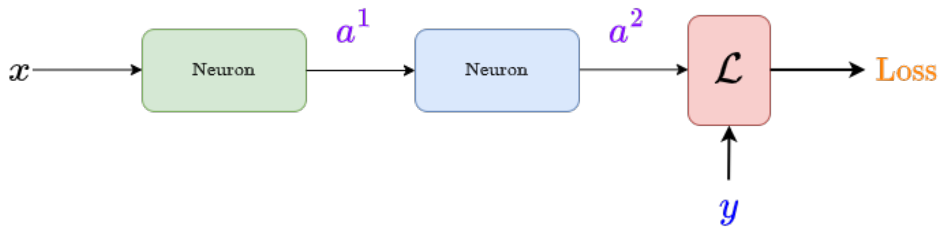
Fall 2022

A two-neuron network: starting backprop

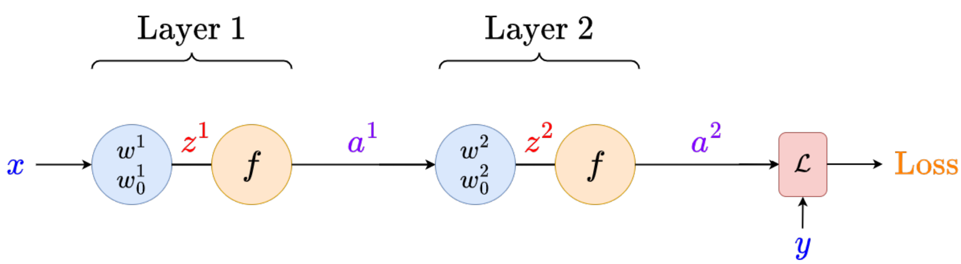
Above, we mention "**each** layer": we'll now transition to a **two-neuron** system, so we have "two layers". Then, we'll build up to many layers.

Remember, though, that the **ideas** represented here are just extensions of what we did **above**.

Let's get a look at our **two-neuron** system, now with our **loss** unit:



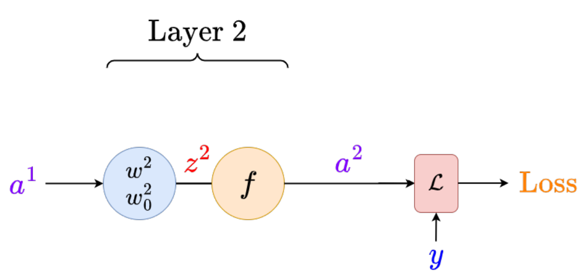
And unpack it:



We want to do **back-propagation** like we did before. This time, we have **two** different layers of weights: w^1 and w^2. Does this cause any problems?

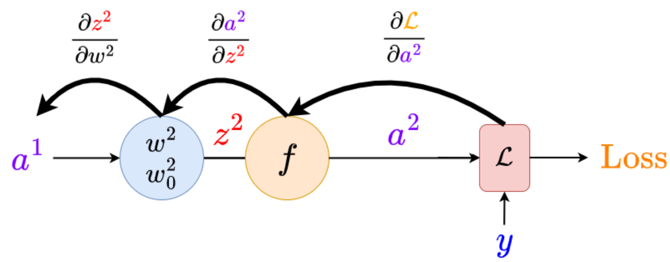
It turns out, it doesn't! We mentioned in the first part of chapter 7 that we can treat the **output** of the **first** layer a^1 as the same as if it were an **input** x.

This is one of the biggest benefits of neural network layers!



Now, we can do backprop safely.

"Backprop" is a common shortening of "back-propagation".



We can get:

$$\frac{\partial \mathcal{L}}{\partial w^2} = \overbrace{\frac{\partial \mathcal{L}}{\partial a^2}}^{\text{Loss unit}} \cdot \overbrace{\frac{\partial a^2}{\partial z^2}}^{\text{Activation}} \cdot \overbrace{\frac{\partial z^2}{\partial w^2}}^{\text{Linear}} \tag{1}$$

The same format as for our **one-neuron** system! We now have a gradient we can update for our **second** weight vector.

But what about our **first** weight vector?

Continuing backprop: One more problem

We need to continue further to reach our **earlier** weights: this is why we have to work **backward**.

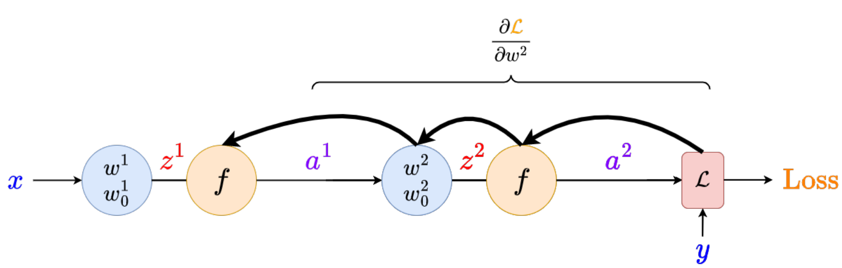
Concept 1

We work **backward** in **back-propagation** because every layer after the **current** one **affects** the gradient.

Our current layer **feeds** into the next layer, which feeds into the layer after that, and so on. So this layer affects **every** later layer, which then affect the loss.

So, to see the effect on the **output**, we have to **start** from the **loss**, and get every layer **between** it and our weight vector.

Remember that when we say "f feeds into g", we mean that the output of f is the input to g.



We have one problem, though:

We just gathered the derivative $\partial\mathcal{L}/\partial w^2$. If we wanted to continue the chain rule, we would expect to add more terms, like:

$$\frac{\partial w^2}{\partial a^1} \tag{2}$$

Since our current derivative includes w^2, we would continue it with a w^2 in the "top" of a derivative,

$$\frac{\partial \mathcal{L}}{\partial w^2}\frac{\partial w^2}{\partial r}$$

We're not sure what "r" is yet.

The problem is, what is w^2? It's a vector of constants.

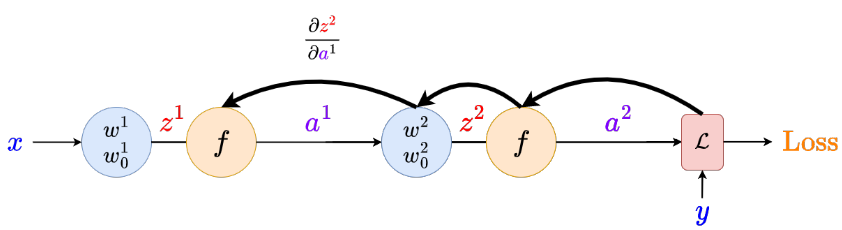
$$w^2 = \begin{bmatrix} w_1^2 \\ w_2^2 \\ \vdots \\ w_n^2 \end{bmatrix}, \qquad \text{Not a function of } a^1! \tag{3}$$

That derivative above is going to be **zero**! In other words, w^2 isn't really the **input** to z^2: it's a **parameter**.

We were building our chain rule by combining inputs with outputs: that's what links two layers together.

So, it should make sense that using something like w (that doesn't link two layers) prevents us from making a longer chain rule.

So, we can't end our derivative with w^2. Instead, we have to use something else. z^2's real input is a^1, so let's go directly to that!



Using this allows us to move from layer 2 to layer 1.

Now, we have our new chain rule:

$$\frac{\partial \mathcal{L}}{\partial a^1} = \overbrace{\frac{\partial \mathcal{L}}{\partial a^2} \cdot \frac{\partial a^2}{\partial z^2}}^{\text{Other terms}} \cdot \overbrace{\frac{\partial z^2}{\partial a^1}}^{\text{Link Layers}} \tag{4}$$

Concept 2

For our **weight gradient** in layer l, we have to end our **chain rule** with

$$\frac{\partial z^\ell}{\partial w^\ell}$$

So we can get

$$\frac{\partial \mathcal{L}}{\partial w^\ell} = \overbrace{\frac{\partial \mathcal{L}}{\partial z^\ell}}^{\text{Other terms}} \cdot \overbrace{\frac{\partial z^\ell}{\partial w^\ell}}^{\text{Get weight grad}}$$

However, because w^l is not the **input** of layer l, we can't use it to find the gradient of **earlier layers**.

Instead, we use

$$\frac{\partial z^\ell}{\partial a^{\ell-1}} \tag{5}$$

To "**link together**" two different layers ℓ and $\ell - 1$ in a **chain rule**.

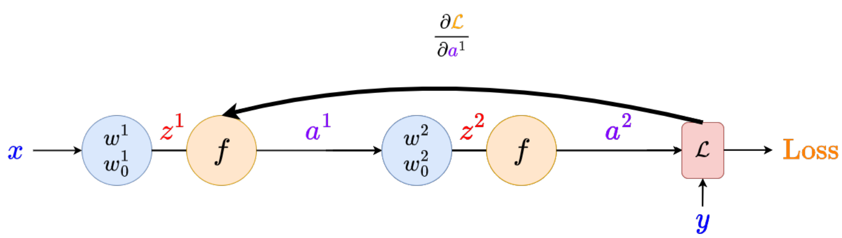
In this section, we compressed lots of derivatives into

$$\frac{\partial \mathcal{L}}{\partial z^\ell}$$

Don't let this alarm you, this just hides our long chain of derivatives!

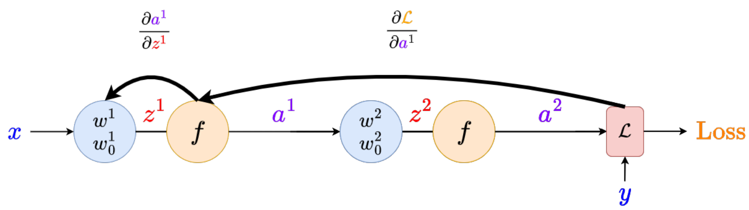
Finishing two-neuron backprop

Now that we have safely connected our layers, we can do the rest of our gradient. First, let's lump together everything we did before:

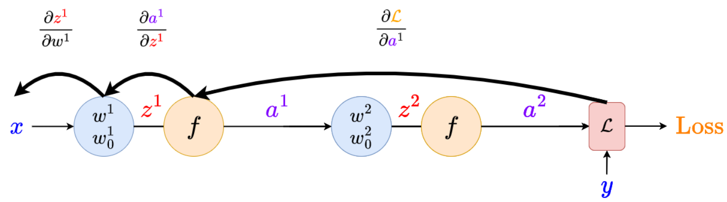


All the info we need is stored in this derivative: it can be written out using our friendly chain rule from earlier.

Now, we can add our remaining terms. It's the same as before: we want to look at the pre-activation



And finally, our input:



We can get our second chain rule

$$\frac{\partial \mathcal{L}}{\partial w^1} = \overbrace{\frac{\partial \mathcal{L}}{\partial a^1}}^{\text{Other layers}} \cdot \overbrace{\frac{\partial a^1}{\partial z^1} \cdot \frac{\partial z^1}{\partial w^1}}^{\text{Layer 1}} \tag{6}$$

Which, in reality, looks much bigger:

$$\frac{\partial \mathcal{L}}{\partial w^1} = \overbrace{\left(\frac{\partial \mathcal{L}}{\partial a^2}\right)}^{\text{Loss unit}} \cdot \overbrace{\left(\frac{\partial a^2}{\partial z^2} \cdot \frac{\partial z^2}{\partial a^1}\right)}^{\text{Layer 2}} \cdot \overbrace{\left(\frac{\partial a^1}{\partial z^1} \cdot \frac{\partial z^1}{\partial w^1}\right)}^{\text{Layer 1}} \tag{7}$$

We see a clear **pattern** here! In fact, this is the procedure we'll use for a neural network with **any** number of layers.

Concept 3

We can get all of our **weight gradients** by repeatedly appending to the **chain rule**.

For each layer, we multiply by

$$\overbrace{\frac{\partial a^\ell}{\partial z^\ell}}^{\text{Within layer}} \cdot \overbrace{\frac{\partial z^\ell}{\partial w^\ell}}^{\text{Get weight grad}}$$

To get the **weight gradient** $\partial \mathcal{L}/\partial w^\ell$.

If we want to **extend** to the next layer, we instead multiply by

$$\overbrace{\frac{\partial a^\ell}{\partial z^\ell}}^{\text{Within layer}} \cdot \overbrace{\frac{\partial z^\ell}{\partial a^{\ell-1}}}^{\text{Link layers}}$$