Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Many neurons per layer

Now, we just have left the elephant in the room: what do we do about the case where we have *full* layers? That is, what if we have **multiple** neurons per layer? This makes this more complex.

Well, the solution is the same as in the first part of chapter 7: we introduce **matrices**.

But this time, with a twist: we have to do **matrix** calculus: a difficult topic indeed.

To handle this, we will go in somewhat **reversed** order, but one that better fits our needs.

- We begin by considering how the chain rule looks when we switch to matrix form.
- We give a general idea of what matrix derivatives look like.
- We list some of the results that matrix calculus gives us, for particular derivatives.
- We actually reason about how matrix calculus *works*.

The last of these is by far the **hardest**, and warrants its own section. Nevertheless, even without it, you can more or less get the idea of what we need - hence why we're going in reversed order.

The chain rule: Matrix form

Let's start with the first: the punchline, how does the chain rule and our gradient descent **change** when we add **matrices**?

It turns out, not much: by using **layers** in the last section, we were able to create a pretty powerful and mathematically **tidy** object.

With layers, each layer feeds into the **next**, with no other interaction. And neurons within the same layer do not directly interact with each other, which simplifies our math greatly.

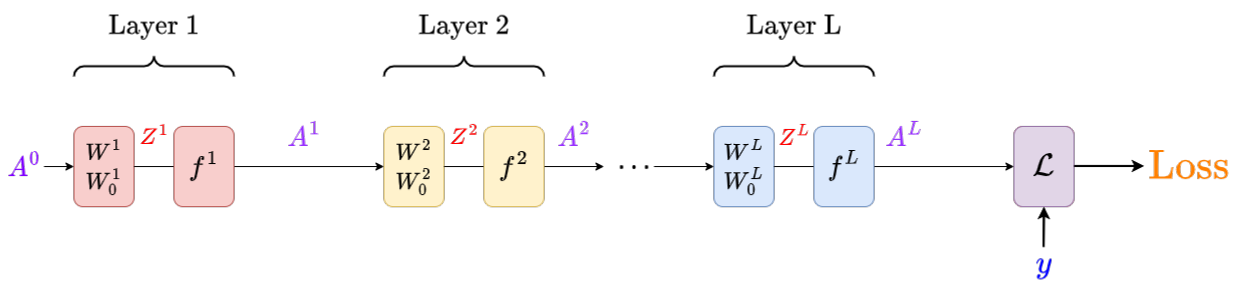
Basically, we have a bunch of functions (neurons) that, within a layer, have nothing to do with each other, and only **output** to the **next** layer of similar functions.

So, we can often **oversimplify** our model by thinking of each layer as like a "big" function, taking in a vector of size m^ℓ and outputting a vector of size n^ℓ.

Our main concern is making sure we have agreement of **dimensions**!

In fact, if you just rearranging your matrices and transposing them can be a helpful way to debug. Be careful, though!

So, here's how our model looks now:



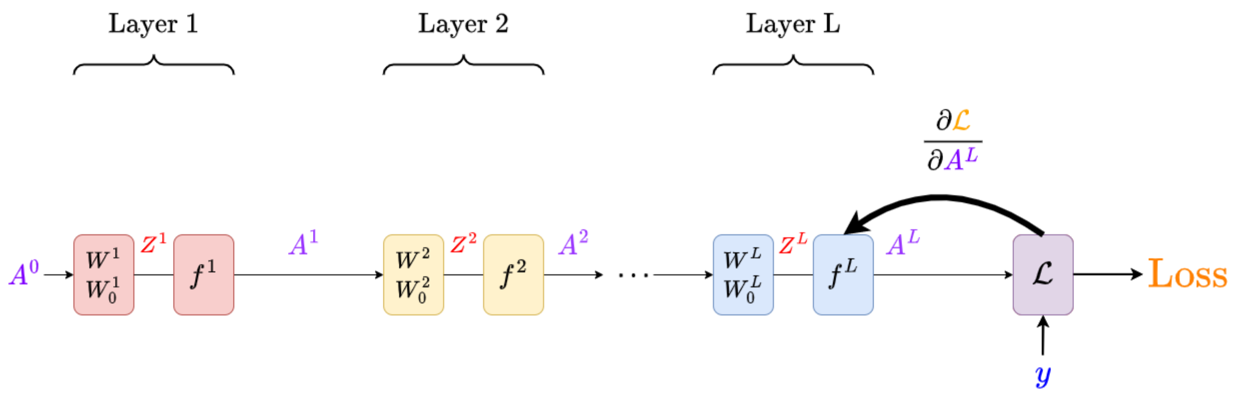
Pretty much the same! Only major difference: swapped scalars for vectors, and vectors for matrices (represented by switching to uppercase)

And, we do backprop the same way, too.

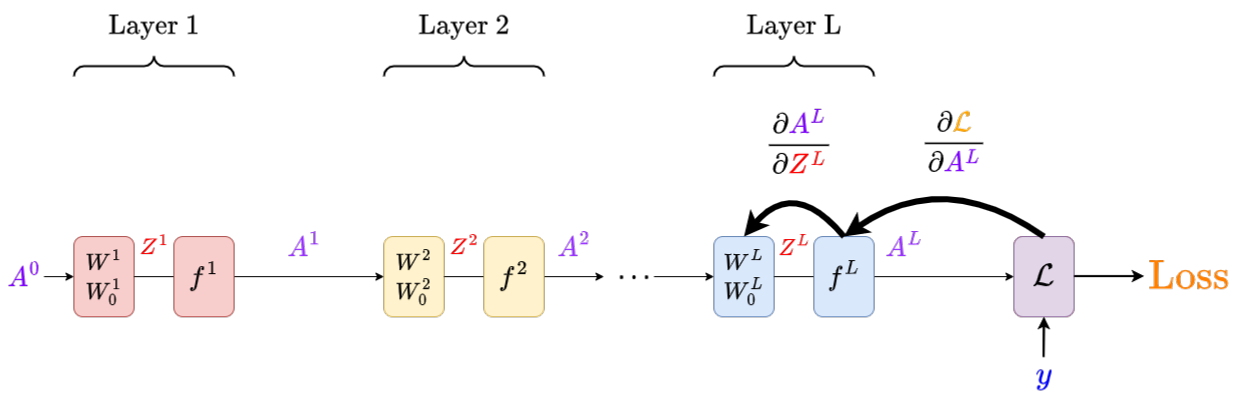
Here, we're not going to explain much as we go: all we're doing is getting the **derivatives** we need for our **chain rule**!

As we go **backwards**, we can build the gradient for each **weight** we come across, in the way we described above.

As always, we start from the loss function:

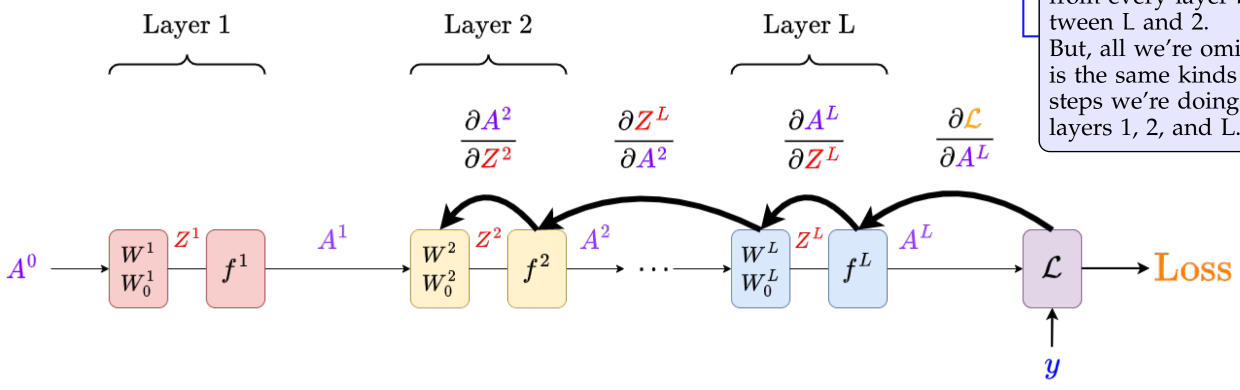


Take another step:

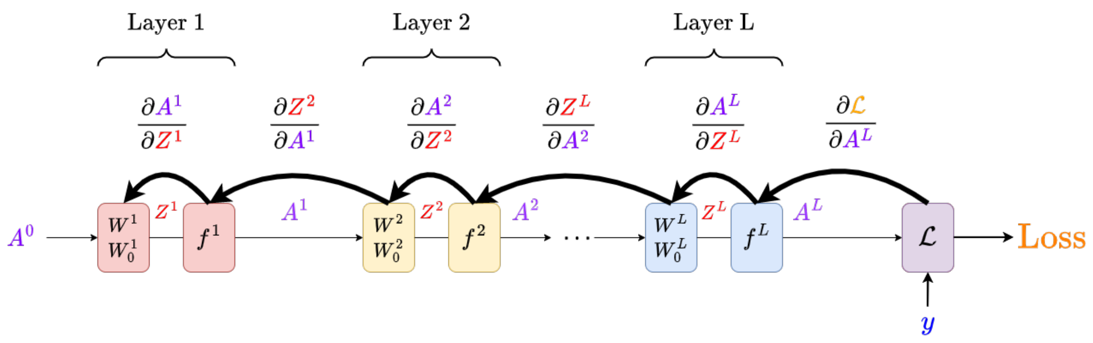


We'll pick up the pace: we'll jump to layer 2 and get its gradient.

The term $\partial Z^L/\partial A^2$ contains lots of derivatives from every layer between L and 2.
But, all we're omitting is the same kinds of steps we're doing in layers 1, 2, and L.


Layer 1
Layer 2
Layer L
$\frac{\partial A^2}{\partial Z^2}$
$\frac{\partial Z^L}{\partial A^2}$
$\frac{\partial A^L}{\partial Z^L}$
$\frac{\partial \mathcal{L}}{\partial A^L}$
A^0
W^1 W^1_0
Z^1
f^1
A^1
W^2 W^2_0
Z^2
f^2
A^2
$\cdots$
W^L W^L_0
Z^L
f^L
A^L
$\mathcal{L}$
Loss
y

Now, we finally get to layer 1!


Layer 1
Layer 2
Layer L
$\frac{\partial A^1}{\partial Z^1}$
$\frac{\partial Z^2}{\partial A^1}$
$\frac{\partial A^2}{\partial Z^2}$
$\frac{\partial Z^L}{\partial A^2}$
$\frac{\partial A^L}{\partial Z^L}$
$\frac{\partial \mathcal{L}}{\partial A^L}$
A^0
W^1 W^1_0
Z^1
f^1
A^1
W^2 W^2_0
Z^2
f^2
A^2
$\cdots$
W^L W^L_0
Z^L
f^L
A^L
$\mathcal{L}$
Loss
y

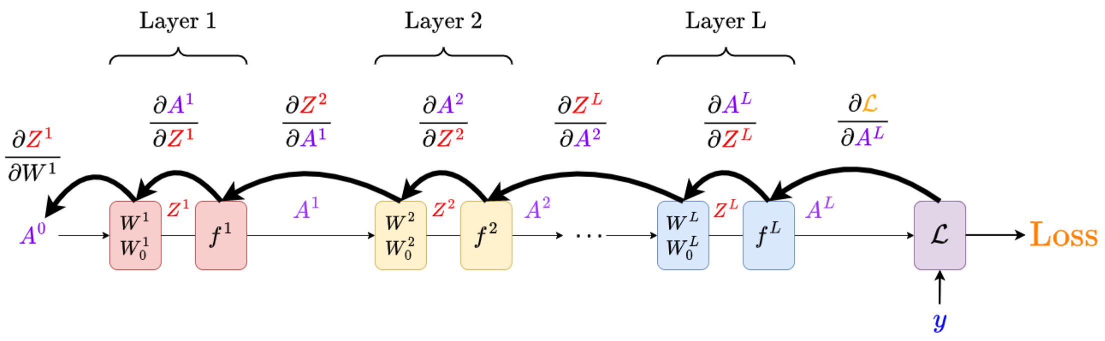
We finish off by getting what we're after: the gradient for W^1.

Notation 1

We depict neural network gradient descent using the below diagram (outside the box):

The **right**-facing **straight** arrows come **first**: they're part of the **forward pass**, where we get all of our values.

The **left**-facing **curved** arrows come **after**: they represent the **back-propagation** of the gradient.


Layer 1
Layer 2
Layer L
$\frac{\partial Z^1}{\partial W^1}$
$\frac{\partial A^1}{\partial Z^1}$
$\frac{\partial Z^2}{\partial A^1}$
$\frac{\partial A^2}{\partial Z^2}$
$\frac{\partial Z^L}{\partial A^2}$
$\frac{\partial A^L}{\partial Z^L}$
$\frac{\partial \mathcal{L}}{\partial A^L}$
A^0
W^1 W_0^1
Z^1
f^1
A^1
W^2 W_0^2
Z^2
f^2
A^2
$\cdots$
W^L W_0^L
Z^L
f^L
A^L
$\mathcal{L}$
Loss
y

And, with this, we can rewrite our general equation for neural network gradients.

How the Chain Rule changes in Matrix form

As we discussed before, we can't just add onto our weight gradient to reach another layer: the final term

$$\frac{\partial Z^\ell}{\partial W^\ell} \tag{1}$$

Ends our chain rule when we add it: W^ℓ isn't part of the input or output, so it doesn't connect to the previous layer.

So, for this section, we'll add it **separately** at the end of our chain rule:

$$\frac{\partial \mathcal{L}}{\partial W^\ell} = \overbrace{\frac{\partial Z^\ell}{\partial W^\ell}}^{\text{Weight link}} \cdot \overbrace{\left(\frac{\partial \mathcal{L}}{\partial Z^\ell}\right)^T}^{\text{Other layers}}$$

That way, we can add onto $\partial \mathcal{L}/\partial Z^\ell$ without worrying about the weight derivative.

Notice two minor changes caused by the switch to matrices:

- The order has to be **reversed**.
- We also have to do some weird **transposing**.

Both of these mostly boil down to trying to be careful about **shape**/dimension agreement.

There are also deeper interpretations, but they aren't worth digging into for now.

Notation 2

The **gradient** $\nabla_{W^\ell}\mathcal{L}$ for a neural network is given as:

$$\frac{\partial \mathcal{L}}{\partial W^\ell} = \overbrace{\frac{\partial Z^\ell}{\partial W^\ell}}^{\text{Weight link}} \cdot \overbrace{\left(\frac{\partial \mathcal{L}}{\partial Z^\ell}\right)^T}^{\text{Other layers}}$$

We get our remaining terms $\partial\mathcal{L}/\partial Z^\ell$ by our usual chain rule:

$$\frac{\partial \mathcal{L}}{\partial Z^\ell} = \overbrace{\left(\frac{\partial A^\ell}{\partial Z^\ell}\right)}^{\text{Layer } \ell} \cdot \left(\cdots\right) \cdot \overbrace{\left(\frac{\partial Z^{L-1}}{\partial A^{L-2}} \cdot \frac{\partial A^{L-1}}{\partial Z^{L-1}}\right)}^{\text{Layer } L-1} \cdot \overbrace{\left(\frac{\partial Z^{L}}{\partial A^{L-1}} \cdot \frac{\partial A^{L}}{\partial Z^{L}}\right)}^{\text{Layer } L} \cdot \overbrace{\left(\frac{\partial \mathcal{L}}{\partial A^L}\right)}^{\text{Loss unit}}$$

This is likely our most important equation in this chapter!