Derivatives of activation functions

(SPEECH)
When

(DESCRIPTION)
Text, One hidden layer, Neural Network. Derivatives of activation functions. Website, deep learning, dot, A.I.

(SPEECH)
you implement back propagation for your neural network, you need to either compute the slope or the derivative of the activation functions.

So, let's take a look at our choices of activation functions and how you can compute the slope of these functions.

(DESCRIPTION)
New slide, Sigmoid activation function.

(SPEECH)
Here's the familiar Sigmoid activation function.

So, for any given value of z, maybe this value of z.

This function will have some slope or some derivative corresponding to, if you draw a little line there, the height over width of this lower triangle here.

So, if g of z is the sigmoid function, then the slope of the function is d, dz g of z, and so we know from calculus that it is the slope of g of x at z.

If you are familiar with calculus and know how to take derivatives, if you take the derivative of the Sigmoid function, it is possible to show that it is equal to this formula.

Again, I'm not going to do the calculus steps, but if you are familiar with calculus, feel free to post a video and try to prove this yourself.

So, this is equal to just g of z, times 1 minus g of z.

So, let's just sanity check that this expression make sense.

First, if z is very large, so say z is equal to 10, then g of z will be close to 1, and so the formula we have on the left tells us that d dz g of z does be close to g of z, which is equal to 1 times 1 minus 1, which is therefore very close to 0.

This isn't the correct because when z is very large, the slope is close to 0.

Conversely, if z is equal to minus 10, so it says well there, then g of z is close to 0.

So, the formula on the left tells us d dz g of z would be close to g of z, which is 0 times 1 minus 0.

So it is also very close to 0, which is correct.

Finally, if z is equal to 0, then g of z is equal to one-half, that's the sigmoid function right here, and so the derivative is equal to one-half times 1 minus one-half, which is equal to one-quarter, and that actually turns out to be the correct value of the derivative or the slope of this function when z is equal to 0.

Finally, just to introduce one more piece of notation, sometimes instead of writing this thing, the shorthand for the derivative is g prime of z.

So, g prime of z in calculus, the little dash on top is called prime, but so g prime of z is a shorthand for the calculus for the derivative of the function of g with respect to the input variable z.

Then in a neural network, we have a equals g of z, equals this, then this formula also simplifies to a times 1 minus a.

So, sometimes in implementation, you might see something like g prime of z equals a times 1 minus a, and that just refers to the observation that g prime, which just means the derivative, is equal to this over here.

The advantage of this formula is that if you've already computed the value for a, then by using this expression, you can very quickly compute the value for the slope for g prime as well.

All right. So, that was the sigmoid activation function.

(DESCRIPTION)
New slide, Tan H activation function.

(SPEECH)
Let's now look at the Tanh activation function.

Similar to what we had previously, the definition of d dz g of z is the slope of g of z at a particular point of z, and if you look at the formula for the hyperbolic tangent function, and if you know calculus, you can take derivatives and show that this simplifies to this formula and using the shorthand we have previously when we call this g prime of z again.

So, if you want you can sanity check that this formula makes sense.

So, for example, if z is equal to 10, Tanh of z will be very close to 1.

This goes from plus 1 to minus 1.

Then g prime of z, according to this formula, would be about 1 minus 1 squared, so there's very close to 0.

So, that was if z is very large, the slope is close to 0.

Conversely, if z is very small, say z is equal to minus 10, then Tanh of z will be close to minus 1, and so g prime of z will be close to 1 minus negative 1 squared.

So, it's close to 1 minus 1, which is also close to 0.

Then finally, if z is equal to 0, then Tanh of z is equal to 0, and then the slope is actually equal to 1, which is actually the slope when z is equal to 0.

So, just to summarize, if a is equal to g of z, so if a is equal to this Tanh of z, then the derivative, g prime of z, is equal to 1 minus a squared.

So, once again, if you've already computed the value of a, you can use this formula to very quickly compute the derivative as well.

(DESCRIPTION)
New slide, Re Lu, and Leaky Re Lu.

(SPEECH)
Finally, here's how you compute the derivatives for the ReLU and Leaky ReLU activation functions.

For the value g of z is equal to max of 0,z, so the derivative is equal to, turns out to be 0 , if z is less than 0 and 1 if z is greater than 0.

It's actually undefined, technically undefined if z is equal to exactly 0.

But if you're implementing this in software, it might not be a 100 percent mathematically correct, but it'll work just fine if z is exactly a 0, if you set the derivative to be equal to 1.

It always had to be 0, it doesn't matter.

If you're an expert in optimization, technically, g prime then becomes what's called a sub-gradient of the activation function g of z, which is why gradient descent still works.

But you can think of it as that, the chance of z being exactly 0.000000.

It's so small that it almost doesn't matter where you set the derivative to be equal to when z is equal to 0.

So, in practice, this is what people implement for the derivative of z.

Finally, if you are training a neural network with a Leaky ReLU activation function, then g of z is going to be max of say 0.01 z, z, and so, g prime of z is equal to 0.01 if z is less than 0 and 1 if z is greater than 0.

Once again, the gradient is technically not defined when z is exactly equal to 0, but if you implement a piece of code that sets the derivative or that sets g prime to either 0.01 or or to 1, either way, it doesn't really matter.

When z is exactly 0, your code will work just.

So, under these formulas, you should either compute the slopes or the derivatives of your activation functions.

Now, we have this building block, you're ready to see how to implement gradient descent for your neural network.

Let's go on to the next video to see that.