Vanishing : Exploding gradients

(SPEECH)
One

(DESCRIPTION)
Text, Setting up your optimization problem. Vanishing, exploding gradients. Website, deep learning, dot, A.I.

(SPEECH)
of the problems of training neural network, especially very deep neural networks, is data vanishing and exploding gradients.

What that means is that when you're training a very deep network your derivatives or your slopes can sometimes get either very, very big or very, very small, maybe even exponentially small, and this makes training difficult.

In this video you see what this problem of exploding and vanishing gradients really means, as well as how you can use careful choices of the random weight initialization to significantly reduce this problem.

(DESCRIPTION)
New slide, Vanishing/exploding gradients.

(SPEECH)
Unless you're training a very deep neural network like this, to save space on the slide, I've drawn it as if you have only two hidden units per layer, but it could be more as well.

But this neural network will have parameters W1, W2, W3 and so on up to WL.

For the sake of simplicity, let's say we're using an activation function G of Z equals Z, so linear activation function.

And let's ignore B, let's say B of L equals zero.

So in that case you can show that the output Y will be WL times WL minus one times WL minus two, dot, dot, dot down to the W3, W2, W1 times X.

But if you want to just check my math, W1 times X is going to be Z1, because B is equal to zero.

So Z1 is equal to, I guess, W1 times X and then plus B which is zero.

But then A1 is equal to G of Z1.

But because we use linear activation function, this is just equal to Z1.

So this first term W1X is equal to A1.

And then by the reasoning you can figure out that W2 times W1 times X is equal to A2, because that's going to be G of Z2, is going to be G of W2 times A1 which you can plug that in here.

So this thing is going to be equal to A2, and then this thing is going to be A3 and so on until the protocol of all these matrices gives you Y-hat, not Y.

Now, let's say that each of you weight matrices WL is just a little bit larger than one times the identity.

So it's 1.5_1.5_0_0.

Technically, the last one has different dimensions so maybe this is just the rest of these weight matrices.

Then Y-hat will be, ignoring this last one with different dimension, this 1.5_0_0_1.5 matrix to the power of L minus 1 times X, because we assume that each one of these matrices is equal to this thing.

It's really 1.5 times the identity matrix, then you end up with this calculation.

And so Y-hat will be essentially 1.5 to the power of L, to the power of L minus 1 times X, and if L was large for very deep neural network, Y-hat will be very large.

In fact, it just grows exponentially, it grows like 1.5 to the number of layers.

And so if you have a very deep neural network, the value of Y will explode.

Now, conversely, if we replace this with 0.5, so something less than 1, then this becomes 0.5 to the power of L.

This matrix becomes 0.5 to the L minus one times X, again ignoring WL.

And so each of your matrices are less than 1, then let's say X1, X2 were one one, then the activations will be one half, one half, one fourth, one fourth, one eighth, one eighth, and so on until this becomes one over two to the L. So the activation values will decrease exponentially as a function of the def, as a function of the number of layers L of the network.

So in the very deep network, the activations end up decreasing exponentially.

So the intuition I hope you can take away from this is that at the weights W, if they're all just a little bit bigger than one or just a little bit bigger than the identity matrix, then with a very deep network the activations can explode.

And if W is just a little bit less than identity.

So this maybe here's 0.9, 0.9, then you have a very deep network, the activations will decrease exponentially.

And even though I went through this argument in terms of activations increasing or decreasing exponentially as a function of L, a similar argument can be used to show that the derivatives or the gradients the computer is going to send will also increase exponentially or decrease exponentially as a function of the number of layers.

With some of the modern neural networks, L equals 150.

Microsoft recently got great results with 152 layer neural network.

But with such a deep neural network, if your activations or gradients increase or decrease exponentially as a function of L, then these values could get really big or really small.

And this makes training difficult, especially if your gradients are exponentially smaller than L, then gradient descent will take tiny little steps.

It will take a long time for gradient descent to learn anything.

To summarize, you've seen how deep networks suffer from the problems of vanishing or exploding gradients.

In fact, for a long time this problem was a huge barrier to training deep neural networks.

It turns out there's a partial solution that doesn't completely solve this problem but it helps a lot which is careful choice of how you initialize the weights.

To see that, let's go to the next video.