Derivatives with a Computation Graph

(DESCRIPTION)
Text, Basics of Neural Network Programming. Derivatives with a Computation Graph. Website, deep learning, dot, A.I.

(SPEECH)
In the last video, we worked through an example of using a computation graph to compute a function J.

Now, let's take a clean diversion of that computation graph.

And show how you can use it to figure out derivative calculations for that function J.

(DESCRIPTION)
New slide, Computing derivatives.

(SPEECH)
So here's a computation graph.

Let's say you want to compute the derivative of J with respect to v.

So what is that?

Well, this says, if we were to take this value of v and change it a little bit, how would the value of J change?

Well, J is defined as 3 times v.

And right now, v = 11.

So if we're to bump up v by a little bit to 11.001, then J, which is 3v, so currently 33, will get bumped up to 33.003.

So here, we've increased v by 0.001.

And the net result of that is that J goes out 3 times as much.

So the derivative of J with respect to v is equal to 3.

Because the increase in J is 3 times the increase in v.

And in fact, this is very analogous to the example we had in the previous video, where we had f(a) = 3a.

And so we then derived that df/da, which with slightly simplified, a slightly sloppy notation, you can write as df/da = 3.

So instead, here we have J = 3v, and so dJ/dv = 3.

With here, J playing the role of f, and v playing the role of a in this previous example that we had from an earlier video.

So indeed, terminology of backpropagation, what we're seeing is that if you want to compute the derivative of this final output variable, which usually is a variable you care most about, with respect to v, then we've done one step of backpropagation.

So we call it one step backwards in this graph.

Now let's look at another example.

What is dJ/da?

In other words, if we bump up the value of a, how does that affect the value of J?

Well, let's go through the example, where now a = 5.

So let's bump it up to 5.001.

The net impact of that is that v, which was a + u, so that was previously 11.

This would get increased to 11.001.

And then we've already seen as above that J now gets bumped up to 33.003.

So what we're seeing is that if you increase a by 0.001, J increases by 0.003.

And by increase a, I mean, you have to take this value of 5 and just plug in a new value.

Then the change to a will propagate to the right of the computation graph so that J ends up being 33.003.

And so the increase to J is 3 times the increase to a.

So that means this derivative is equal to 3.

And one way to break this down is to say that if you change a, then that will change v.

And through changing v, that would change J.

And so the net change to the value of J when you bump up the value, when you nudge the value of a up a little bit, is that, First, by changing a, you end up increasing v.

Well, how much does v increase?

It is increased by an amount that's determined by dv/da.

And then the change in v will cause the value of J to also increase.

So in calculus, this is actually called the chain rule that if a affects v, affects J, then the amounts that J changes when you nudge a is the product of how much v changes when you nudge a times how much J changes when you nudge v.

So in calculus, again, this is called the chain rule.

And what we saw from this calculation is that if you increase a by 0.001, v changes by the same amount.

So dv/da = 1.

So in fact, if you plug in what we have wrapped up previously, dv/dJ = 3 and dv/da = 1.

So the product of these 3 times 1, that actually gives you the correct value that dJ/da = 3.

So this little illustration shows hows by having computed dJ/dv, that is, derivative with respect to this variable, it can then help you to compute dJ/da.

And so that's another step of this backward calculation.

I just want to introduce one more new notational convention.

Which is that when you're witting codes to implement backpropagation, there will usually be some final output variable that you really care about.

So a final output variable that you really care about or that you want to optimize.

And in this case, this final output variable is J.

It's really the last node in your computation graph.

And so a lot of computations will be trying to compute the derivative of that final output variable.

So d of this final output variable with respect to some other variable.

Then we just call that dvar.

So a lot of the computations you have will be to compute the derivative of the final output variable, J in this case, with various intermediate variables, such as a, b, c, u or v.

And when you implement this in software, what do you call this variable name?

One thing you could do is in Python, you could give us a very long variable name like dFinalOurputVar/dvar.

But that's a very long variable name.

You could call this, I guess, dJdvar.

But because you're always taking derivatives with respect to dJ, with respect to this final output variable, I'm going to introduce a new notation.

Where, in code, when you're computing this thing in the code you write, we're just going to use the variable name dvar in order to represent that quantity.

So dvar in a code you write will represent the derivative of the final output variable you care about such as J.

Well, sometimes, the last l with respect to the various intermediate quantities you're computing in your code.

So this thing here in your code, you use dv to denote this value.

So dv would be equal to 3.

And your code, you represent this as da, which is we also figured out to be equal to 3.

So we've done backpropagation partially through this computation graph.

Let's go through the rest of this example on the next slide.

So let's go to a cleaned up copy of the computation graph.

And just to recap, what we've done so far is go backward here and figured out that dv = 3.

And again, the definition of dv, that's just a variable name, where the code is really dJ/dv.

We've figured out that da = 3.

And again, da is the variable name in your code and that's really the value dJ/da.

And we hand wave how we've gone backwards on these two edges like so.

Now let's keep computing derivatives.

Now let's look at the value u.

So what is dJ/du?

Well, through a similar calculation as what we did before and then we start off with u = 6.

If you bump up u to 6.001, then v, which is previously 11, goes up to 11.001.

And so J goes from 33 to 33.003.

And so the increase in J is 3x, so this is equal.

And the analysis for u is very similar to the analysis we did for a.

This is actually computed as dJ/dv times dv/du, where this we had already figured out was 3.

And this turns out to be equal to 1.

So we've gone up one more step of backpropagation.

We end up computing that du is also equal to 3.

And du is, of course, just this dJ/du.

Now we just step through one last example in detail.

So what is dJ/db?

So here, imagine if you are allowed to change the value of b.

And you want to tweak b a little bit in order to minimize or maximize the value of J.

So what is the derivative or what's the slope of this function J when you change the value of b a little bit?

It turns out that using the chain rule for calculus, this can be written as the product of two things.

This dJ/du times du/db.

And the reasoning is if you change b a little bit, so b = 3 to, say, 3.001.

The way that it will affect J is it will first affect u.

So how much does it affect u?

Well, u is defined as b times c.

So this will go from 6, when b = 3, to now 6.002 because c = 2 in our example here.

And so this tells us that du/db = 2.

Because when you bump up b by 0.001, u increases twice as much.

So du/db, this is equal to 2.

And now, we know that u has gone up twice as much as b has gone up.

Well, what is dJ/du?

We've already figured out that this is equal to 3.

And so by multiplying these two out, we find that dJ/db = 6.

And again, here's the reasoning for the second part of the argument.

Which is we want to know when u goes up by 0.002, how does that affect J?

The fact that dJ/du = 3, that tells us that when u goes up by 0.002, J goes up 3 times as much.

So J should go up by 0.006.

So this comes from the fact that dJ/du = 3.

And if you check the math in detail, you will find that if b becomes 3.001, then u becomes 6.002, v becomes 11.002.

So that's a + u, so that's 5 + u.

And then J, which is equal to 3 times v, that ends up being equal to 33.006.

And so that's how you get that dJ/db = 6.

And to fill that in, this is if we go backwards, so this is db = 6.

And db really is the Python code variable name for dJ/db.

And I won't go through the last example in great detail.

But it turns out that if you also compute out dJ, this turns out to be dJ/du times du.

And this turns out to be 9, this turns out to be 3 times 3.

I won't go through that example in detail.

So through this last step, it is possible to derive that dc is equal to.

So the key takeaway from this video, from this example, is that when computing derivatives and computing all of these derivatives, the most efficient way to do so is through a right to left computation following the direction of the red arrows.

And in particular, we'll first compute the derivative with respect to v.

And then that becomes useful for computing the derivative with respect to a and the derivative with respect to u.

And then the derivative with respect to u, for example, this term over here and this term over here.

Those in turn become useful for computing the derivative with respect to b and the derivative with respect to c.

So that was the computation graph and how does a forward or left to right calculation to compute the cost function such as J that you might want to optimize.

And a backwards or a right to left calculation to compute derivatives.

If you're not familiar with calculus or the chain rule, I know some of those details, but they've gone by really quickly.

But if you didn't follow all the details, don't worry about it.

In the next video, we'll go over this again in the context of logistic regression.

And show you exactly what you need to do in order to implement the computations you need to compute the derivatives of the logistic regression model.