MITOCW | OCW_18.S096_Lecture05-Part2_2023jan27.mp4

PROFESSOR: So you've already seen a little bit of the story of forward mode and reverse mode from Stephen last week. One version of the story is that you're multiplying derivatives, or Jacobian matrices, or something like that.

And, of course, you've heard Stephen say that matrix multiplication is associative, and so you can go left to right or right to left, but it matters what order you go in terms of the complexity of the computation. That one order might be an n cubed computation, and another order might be an n squared computation. And so you saw an example of that.

And in some fundamental sense, that describes the entire story of forward and reverse mode. But in a way, I feel like it hides more than it reveals. And the story is-- in some sense, the entire story can be reduced to that. But I feel like that's not enough to fully understand.

And so I put together this example that I used in my class last semester. And I'm just going to pull it all out. I'm just going to grab a simple example from calculus and show you what's really going on.

And so I want to take this simple example. There's nothing special about it. I just randomly came up with it, where I'm going to just input an x and a y. And I'm going to have three lines of code. So to speak, where three computations that are going to happen.

I'm going to take a to be sine x-- no reason whatsoever. I'm going to take to b to the a, divided by y. And then I like x and y going in, and z, being the last letter, being the final output. So z, I'll have it be b plus x.

You could see how that kind of looks like a computer program. It feels more like a computer program than mathematics, where you're writing an equality. It looks like a sequence of steps, where at every step, you at least-- you have the variables that came before.

What is a computer program in the end? It's a formula, where on the right-hand side, you know the value of everything. And so on the left-hand side, you can define the thing. That's what a computer program basically is. And one could have a problem, like one could create this problem, which is, say, find the derivatives. Like find dz, dx, or find dz, dy.

And this is pretty simple. You all know how to do it. Let's just-- let's just start with the basics. So how might we do this? Well, let's see.

So z is b plus x. Let's just figure out what's going on here. B is a over y. So this is a over y plus x. Because you want to get everything in terms of the x's and y's. A is sine x. So we have sine x over y, plus x.

And then, now that we have everything in terms of x and y's-- we're all very good at this-- dz, bx, of course-- the actual answer is cosine x over y, plus 1. And then dz, by is-- what is it? It's a minus sine x over y squared.

No controversy to this. All simple stuff. If my colleagues caught me doing this to you advanced students, they would make fun of me. This is just baby stuff. But let's establish a little bit-- let's take a good look at what we just did. Let's take a close look at this sort of thing.

And let me introduce a computational graph. Let me try to draw a picture of the computation we just did with a computational graph. So let's write that-- computational graph. And by the way, these notes are online. I'll put a pointer up. It's a terrible handwritten version. One day I'll type this up better.

But we'll have a computational graph. The graph will be a DAG, if you know what that word means-- Directed Acyclic Graph, which basically means that there are arrows on all the edges and there are no cycles. That's what a computer program is. It's how do you build a next variable from a previous variable. And if you ever look leftward, all the data is available to you so that you don't get an error.

And so I like to-- people are not completely standard as to how they draw computational graphs. It drives me crazy. I'm going to take the convention that I'm going to put the variables at-- the variable names as nodes.

So my input nodes are x and y. And let's see, if I look at step one over there, the first thing I'm going to calculate is a. So that's going to be a vertex or a node. And while I'm at it, I'm going to draw this arrow right here.

And what I'm going to put on that arrow is not-- I'm not going to put the-- you could draw the computation, the sine, but what I'm going to do is actually put the derivative on the arrow. And so I'm going-- on the arrow, I'm going to put the function, cosine x. So the derivative of-- oh, it's just this. The derivative-- the way to read this is the derivative of a with respect to x is what's on this arrow. So this is ba, bx.

So let's go another step. B is a over y. So I think you've got the idea now that b going to be my node. And I'm going to calculate db, da. What is db, da? What should I put here as the function? I want db, da. I want the derivative-- I always want the one-step derivative between this variable and this variable. So I want the-- if I vary a little bit, what do I multiply by? What's db, da?

STUDENT: [INAUDIBLE]

PROFESSOR: 1 over y-- good. Right. That's just-- I started with a over y, and I think we're going to switch to a. So that's 1 over y. Good. But we also have a db, dy. So let's put an arrow like that. And what's db, dy? Again, simple questions, but got to keep you guys awake, keep me awake. What's db, dy?

STUDENT: Negative a over y squared.

PROFESSOR: Negative a over y squared. Good. And finally, we have a z. And z depends on b. And it also depends on f. So we have a dependence that goes back to the beginning with z. And so let's see, what are dz, db, and dz, bx? They're kind of the same answer to both. What's dz, db, and what's dz, bx? Come on, first grade question, really.

STUDENT: 1?

PROFESSOR: Right, they're both 1. Dz, db is 1. Db, zx is one. Because z is just b plus x. So derivatives on the edges-- you get the point that the derivative is labeled on the edges. Derivatives on edges, just to write that down for you. And it's just-- I like to think of this as a one-step derivative.

So it's like-- it's a derivative of one line of code, if you like. I'm not putting in the-- I'm not putting in the full long-range derivative. I'm just putting in the one-step derivative. So in other words, I'm not putting in this thing, which is the full derivative. It's just the one step that I'm putting in.

I wanted to put on the graph what we just did to-- well, let's get the answer now. So I claim that one way to get the actual answer is to think of it graphically, that you could start over here, at x, and we want to head to z. And we're going to look at all the paths that will take us from x to z. There's one path that goes like this. And then there's another path that goes like this.

So there's two paths that'll take us from x to z. And what I'd like to do is, basically, walk along the path and then write down the derivative I see as I go. And I'm going to write it-- I'm going to write it right to left. So let me start walking from x to a along this path.

When I go from x to a, I pick up a cosine x. So this is step one. I pick up a cosine x. Then I have to step over from a to b. So I pick up 1 over y. So that's my step two. And then finally, when I go from b to z, I have a factor of 1. So that's my step three.

I have another path I have to cross. I have to take all possible paths. So my next path is the one that goes from x to z. And so I add the 1 over here. There's only one step to that. And so that's the answer, actually-- 1 over y cosine x plus 1 is the answer for derivative z with respect to x.

So you can view it in that way as all possible paths, from input to output. And then just multiply as you go. And, of course, with scalars, I could have multiplied in any order. But you can imagine-- I hope you can understand why I went from right to left. I didn't really need it for this problem.

But I wanted to set up a good plan for when these are not scalars, but these are vectors or matrix valued functions. And then the order matters. And so the matrix multiply has to go from right to left. In this case, it wouldn't have mattered. So this is the correct answer for db, zx.

And dz, dy, similarly-- at the first step, we have minus a over y squared. That's step one. Step two is to multiply that by 1, which doesn't do anything. And you see, the answer is-- what is the answer? The answer is minus-- minus a over y squared, which you could substitute. The computer wouldn't care. If the computer-- the computer has the value of a. And a is sine x. But you might like to see it in that format.

So this is forward mode, automatic differentiation. This is basically what was going on in the algorithm that I just showed you with the Babylonian algorithm. This is maybe the better way to look at it, where what's happening is as you traverse through the computer program, in that order, you can actually calculate each of these things in order as well. And, thereby, you can actually accumulate the derivatives as you go.

So this is the forward mode view of differentiations. And like I said, there's nothing magic about everything being a scalar here. Every one of these could be a function, like you've seen in this class. For example, it could have been that x was a matrix and a was the square function or the inverse function of a matrix.

This could have been a determinant or this could have been a matrix and this could have been a determinant. And then in this case, you've got the gradient of the determinant, with respect to the matrix, the very thing I showed you earlier, with the aggregate.

So the only thing that's required is the associativity. And the only thing that matters is that if you ever bring things together, you have to add the answers. So you can imagine a computer program where there's all sorts of arrows coming from left to right. And as long as more than one arrow comes in, you just add the answers. Because that's how derivatives work. So that's forward mode of differentiation.

There is a backward mode where you follow the paths backwards. So when you follow it backwards-- so let me just see. Here's where-- OK, I'm not going to transpose it. Here's where I'm actually using-- I'm going to use the fact that it's scalars now.

So this whole calculation I just showed you was forward. So this and this is forward mode. I'm going to reverse modes to scalars. When we get to matrices, we might have to transpose things. But let me just show you reverse mode for scalars just to get that correct.

So reverse mode for scalars says, OK, let's start not on the left end, but let's start on the right end. And you might remember a week ago, I said when had sine of x squared-- how many of you-- I asked the question, how many of you would-- the derivative would be the cosine of x squared times 2x, and how many would have said 2x times the cosine of x squared? It's a matter of going inside out or outside in. And you can go either way.

So for the reverse mode, what we're going to do is we're going to follow our way from the z to the x. And, of course, there's two ways to do that. And if you do that-- I'm going to, again, write it from right to left. I'm going to start-- I'll take that first-- that horizontal path.

And I'm just going to go-- I'm going to write down the one, as the first thing I do. And then the second thing I'm going to do is write down the 1 over y. And then the third thing I'm going to do is write down the cosine x.

But every time, when I go right to left, when the path splits like that, I also have to add it. So I'm also going to have to add a 1 as well. And I'll do that on the-- I don't know when you're going to do that, but I'll just say you can do that on step one as well.

And so that's-- and here, again, you're going to do the 1 on the first step. And the minus a over y squared we're now going to do in the second step. And either way, we're going to get the solution to dz, bx dz, by.

And in a sense, every calculation in the world can be looked at as a DAG. And it could be looked at as operations. And you could think of it as basically following paths like this.

So to emphasize this, in a way, you can embed all this in matrices, but I feel like it hides. Without seeing the graph structure, you don't really get the full feel, I think, of what-- oh, yes?

STUDENT: I was wondering, I don't know if you're recording the [INAUDIBLE].

PROFESSOR: Oh, my gosh. I don't know-- yeah, good point. I forgot to put on the mic. Thank you for catching that. I don't know how well it will work, probably badly. Were you able to hear me, Stephen? Maybe the Zoom recording is not so bad.

AUDIENCE: I can hear you.

PROFESSOR: So we actually have a backup if we know how to splice it in. But I'm going to put it on now. Thank you for catching that. Any questions about non-technical stuff, but forward and reverse-- not the audio visual stuff?

So let's delve in a little bit about how does one think about this. So there's a graph theory way and an implementation way of thinking about this a little bit. So the graph theory way of thinking about this is to think about the fact that what we want to do is really calculate the sum of all the path products from inputs to outputs. So I just gave you a term.

I'm going to define a path product. I'll define it loosely. I hope this will be good enough. The path product will be the product of the edge weights. The product has to be in the right order if it's associative, but not commutative. But product of edge weights as you traverse a path.

So the path products are-- so cosine x over y is one path product, with that length 3 path. One is just that length 1 path. And so those are the two path products. And then what we're interested in is, one way or another, calculating the sum of path products from inputs to outputs.

That's kind of the real goal of doing that. And it doesn't really matter whether you go from the end of the path and move your way to the other end, or if you start at the beginning of the path and go to the end. And so when you see it that way, I think reverse mode and forward mode don't seem so mysterious.

I think it's pretty clear, if you stand back here, the world doesn't care if you go cosine x times 1 over y, times 1, or if you go 1 times 1 over y, times cosine x. And the only issue is if these were matrices, how would you do it?

But I think you all get the idea that if the path products-- like, if this was A, B, and C-- if these were-- I'm using capital letters to have matrices-- then the world doesn't really care if you were to calculate-- if you traverse this way, and you saw the A first.

And then when you multiply by B, and then you see the C last, or if you went backwards, and you pick up the C first, and you then multiply by B on the left, and then finally, A on the left.

As long as you have an associative system, right it doesn't matter which way you do those multiplies. So as long as you can traverse the paths, either from forward to back, or back to forward. Or by the way, you can-- not that this happens very much, but you could even traverse paths from the middle outward.

And as long as you put the right things in the right order, there's no rule of the universe that says you have to go from the beginning to the end or the end to the beginning, or you can't go middle outward. The beauty of associativity is these path products will work just any which way you do it.

So that's one way to look at automatic differentiation, both forward and reverse, is to think of it in terms of these path products that really don't care how you go. Now as far as implementations are concerned-- so one thing I'll do is I'll move the laptop over a little bit so that Stephen has a chance of seeing what we're doing. It might be a weird angle. But you know what, let's do a little better. Let's move the whole laptop so Stephen can see it a little bit better if I use these boards here.

So let's-- so let's take a little bit of a closer view of implementation of forward mode now that we have this understanding of what it is we're trying to do. So how would we implement this thing? So how are we going to do this?

Well, let me focus on being in the middle. So suppose we have a lot of stuff have happened. I don't know what's going on. And we're at the point in time where we're here, and what we want to do-- let me call this-- I'll call this x. Just it's not an input at the beginning. It's just x is somewhere in the middle. And we're going to calculate an output. We're going to calculate f of x.

And so whatever comes in here, we're going to somehow have it go-- we're taking a path product. And so we've come up to here. And if you just kind think-- I don't know, is this recursive thinking? Or is this just how one should think about any computer program?

But one way or another, you've gotten to here. And you have the path product up until here. So we start out, like maybe we call it an inductive hypothesis or whatever you'd like to say. We know the path product up to here. So if I knew the path product up to here, and I going forward mode, what's the next path product?

Suppose I know this path product. Let's call it P for product. What's the path product here?

STUDENT: [INAUDIBLE]

PROFESSOR: Exactly. It's just-- what is it the that?

STUDENT: F prime.

PROFESSOR: Right, f prime, times the previous-- exactly. So one way or another, I need a data structure that-- I need a data structure that will take the value here and the path product-- the path product, and it'll give me-- it'll give me f of the value if I want to run-- if I just want to run the algorithm-- and I need the path product times f prime. And I want to multiply in this order.

And so in some sense, this is what the whole dual number system thing is doing. This is another way to look at the dual numbers. So there's lots and lots of ways to understand what I just showed you with dual numbers. But it's really nothing more than taking your path products with the value.

And you could see-- the reason why I'm showing you this way is that if you're executing a computer program, where you want to-- literally, you want-- the main thing the computer program was meant to do is to calculate f of x. And now this additional extra thing we want it to do-- maybe we're doing gradient descent, and machine learning, or who knows what we're doing. We want the derivative to happen at the same time.

Then all we have to do is overload our program that was already happy to calculate the f of the value, and then tack along the path product, and append the path product with this extra multiply. And so this is how-- this is one way of looking as to how the whole dual number thing is actually working.

And so in symbols, if I had x, comma, p, then on the next step, I'm going to have f of x, and f prime of x, times P. And so that's how we can carry forward this path product idea.

And now let's talk about how to start this whole thing. So this was in the middle. How should we start? So x, comma-- what should I put here so that the first step just works? Is it obvious what we should just start with? Or another way to say it is, what should be-- what should be the path product of a path of length 0?

Do people ever think about these things? Like, if I asked you what is the sum of the empty vector, and I told you, there's only one right answer to that, what's the right answer of the sum of an empty vector, a vector of length 0? 0-- it's the identity element. It's the only thing that makes sense.

Why? Because you always want the sum from I equals 1 to k plus 1 of xi to equal xk plus 1, plus the sum from I equals 1 to k. And if you make this work for k equals 0, it's perfect. And what's the empty product? It's 1, right? Determinant of a 0 by 0 matrix also should be 1. And then the Laplace expansion works.

So what's the empty path product then? So how should we-- in effect, how-- when you sum, if you're summing up a vector, you initialize the variable to 0, and then you start adding in numbers. So what's the empty-- what's the empty start? 1, exactly. So there are a number of ways to interpret this. You could think of this as the slope being-- but this is not a bad way to think about it. Again, you could think of it in multiple ways.

But you start with x, comma, 1 to see the operation. And then at every step, you do this. And at the very end, you get the derivatives that you're looking for. So that's forward mode. And it works just great.

A quick check, though. Suppose I had f of x as a constant, like 2. And then, so I feed it in x, comma P, what's the output? What are the two numbers that would come out if anywhere in the middle of my calculation, I started with xP and I applied this constant function? 2, 0. It doesn't even matter what the input is. That's right, because it's a constant function. It doesn't care.

So this is the one arrow case in forward mode. Maybe it's worthwhile to quickly talk about the multiple arrow case. So for example, suppose I have-- let's say I had ap and bq. And let's say I had two arrows going in. And I had z is some function of two variables, like I did over there, maybe the sum, or the product, or divide, or any function of two variables.

So this will be the one step derivative, of course. This will be dz, da. And this will be-- I don't know what the best way to write this is, but dz, da, dz, db is probably as good a way as any. So now I'm not thinking of a and b as numbers, but I'm thinking of them as symbols for the moment. Or I'm imagining that one way or another, I know the derivative of this-- at least one way or another, I know the derivative of this function with respect to this variable somehow.

Just to give you a quick example that this is not so complicated. If f was the plus function-- if this is a plus b, what would I put here and here? Again, simple question. 1 and 1. And slightly more complicated, but not by much, if this was the product function, a times b, what would I put here and what would I put here?

STUDENT: [INAUDIBLE]

PROFESSOR: Say that again.

STUDENT: B and a?

B and a, perfect. So the point is for lots of basic functions, it's very easy to know what to put here. And so you've got these multiple arrows going in. And-- yeah, you have these multiple arrows in, and, of course, what we're just going to do is we're just going to add the results. And so did I write it out? In effect, yes.

So let me just say that if this goes in, what we really want to come out is the z, which is, of course, f of a and b. And then what we want to do is to continue the paths-- let's see, we had-- did I write this correctly? Let me get this right. So let's see.

So if I started with this, then what-- oh, yeah, what I have to do is take-- is this right? I have to take P-- yeah, p times dz, za, plus q times-- yeah, that's right-- dz, db. And that's the right thing to do to carry forward the derivative. And so this is the-- because this is exactly what the derivative would be over here.

Or if you like, you could just think of this as combining the paths, the path products. So you could think of this from a calculus point of view, that the derivative-- so the calculus viewpoint is that the derivative of this, with respect to that, plus the derivative of this, either the first variable, then times the second variable.

But the pathway, which I think is almost easier to-- it all depends on whether you like to think calculus first or you like to think paths first. But it's really just different words for what in mathematics is the same thing, that we're taking this path product and this path product. And the rule is that anytime things come together, just like the one you saw here, you just add them.

OK, well, I've run out of time. I've got a whole bunch of more notes on reverse mode, on how do you do the same thing with reverse mode. But I don't know whether-- I'm not going to see you before next week today. So you might see some version of this from Chris, or from Gaurav, or maybe from Stephen. Or otherwise, I'll give you my own version anyway by the end next week.

All right, so I'll wish you all a good weekend. And you'll be in good hands next week.