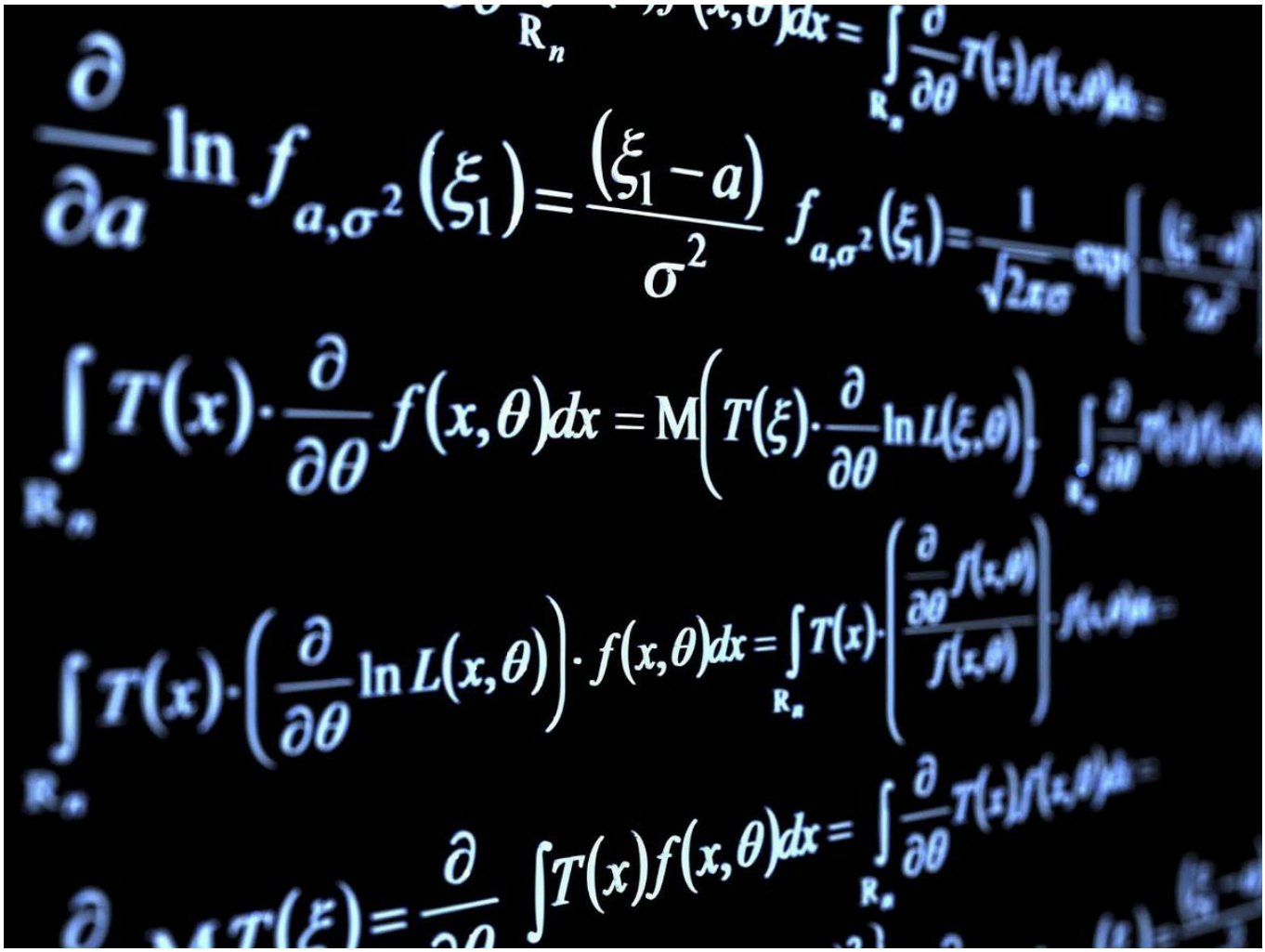
Automatic Differentiation, Explained

How Do Computers Calculate Derivatives?

Chi-Feng Wang Follow
Mar 3 · 3 min read ★



Title image: Source

Neural networks are able to gradually increase accuracy with every training session through the process of gradient descent. In gradient descent, we aim to minimize the loss (i.e. how inaccurate the model is) through tweaking the weights and biases.

As explained in a previous series, by finding the partial derivative of the loss function, we know how much (and in what direction) we must adjust our weights and biases to decrease loss. In that series, we calculated the derivative mean squared error loss function of a single-neuron neural network by hand.

However, how do neural networks—computers—calculate the partial derivatives of an expression? The answer lies in a process known as **automatic differentiation**. Let me illustrate it to you using the cost function from the previous series, but tweaked so that it's in scalar form.

$$C(y, w, x, b) = y - max(0, w \cdot x + b)$$

Image 1: The cost function in scalar form

In addition, because automatic differentiation can only calculate the partial derivative of an expression on a certain point, we have to assign initial values to each of the variables. Let us say: y=5; w=2; x=1; and b=1.

Let's find the derivative of the function!

. . .

Before we can begin deriving the expression, it must be converted into a computational graph. A computational graph simply turns each operation into a **node** and connects them through lines, called **edges**. The computation graph for our example function is shown below.

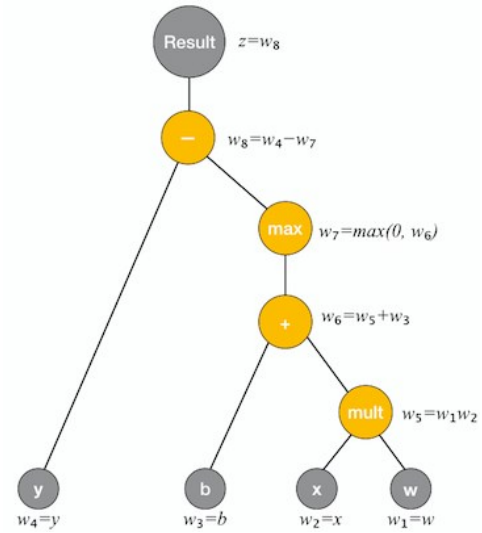


Figure 2: Computation graph for the scalar cost function

First, let us calculate the values of each node, propagating from the bottom (the input variables) to the top (the output function). This is what we get:

Node	Expression	Value
w_1	w	2
w_2	x	1
w_3	b	1
w_4	y	5
w_5	$w_1 \cdot w_2$	2
w_6	$w_5 + w_3$	3
w_7	$\max(0, w_6)$	3
w_8	$w_4 - w_7$	2
z	w_8	2

Figure 3: Values of each node

Next, we need to calculate the partial derivatives of each connection between operations, represented by the edges. These are the calculations of the partials of each edge:

$\frac{\delta w_5}{\delta w_1} = \frac{\delta w_1 w_2}{\delta w_1} = w_2$	$\frac{\delta w_5}{\delta w_2} = \frac{\delta w_1 w_2}{\delta w_2} = w_1$
$\frac{\delta w_6}{\delta w_5} = \frac{\delta w_5 + w_3}{\delta w_5} = 1$	$\frac{\delta w_6}{\delta w_3} = \frac{\delta w_5 + w_3}{\delta w_3} = 1$
$\frac{\delta w_7}{\delta w_6} = \frac{\delta \max(0, w_6)}{\delta w_6} = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$	$\frac{\delta w_8}{\delta w_7} = \frac{\delta w_4 - w_7}{\delta w_7} = -1$
$\frac{\delta w_8}{\delta w_4} = \frac{\delta w_4 - w_7}{\delta w_4} = 1$	$\frac{\delta z}{\delta w_8} = 1$

Figure 4: Values of each edge

Notice how the partial of the max(0, x) piece-wise function is also a piece-wise function. The function converts all negative values to zero, and keeps all positive values as they are. The partial of the function (or graphically, its slope), should be clear on its graph:

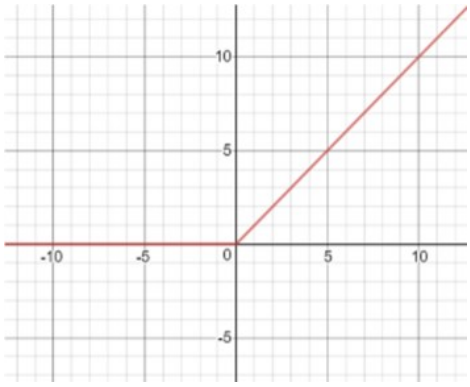


Figure 5: max(0, x) function. As seen, there is a slope of 1 when x>0, and a slope of 0 when x<0. The slope is undefined when x=0.

Now we can move on to calculating the partials! Let's find the partial of with respect to the weights. As seen in Figure 6, there is just one line that connects the result to the weights.

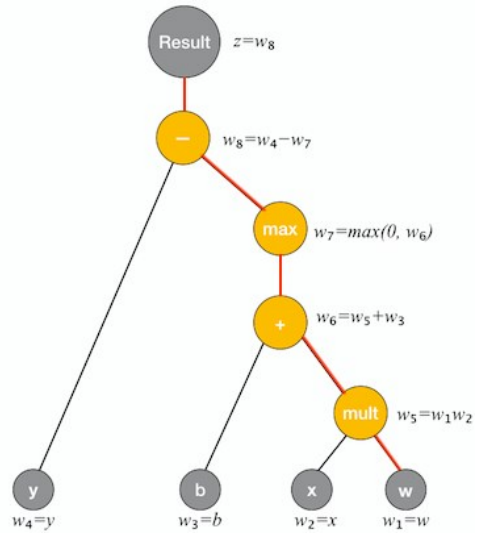


Figure 6: The path that connects the function to the weights

Now we simply multiply up the edges:

$$\frac{\delta z}{\delta w_1} = \frac{\delta z}{\delta w_8} \times \frac{\delta w_8}{\delta w_7} \times \frac{\delta w_7}{\delta w_6} \times \frac{\delta w_6}{\delta w_5} \times \frac{\delta w_5}{\delta w_1} = 1 \times (-1) \times \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases} \times 1 \times w_2 = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$$

Figure 7: Partial

And that's our partial!

This whole process can be completed automatically, and allows computers to compute the partial derivative of a value of a function accurately and quickly. It is this process that allows AI to be as efficient as it is today.

. . .

If you like this article, or have any questions or suggestions, feel free to leave a comment below!

Automatic Differentiation | Deep Learning | Derivatives | Step By Step | Differentiation

182 claps

Chi-Feng Wang

Intern at Augentix Inc. // Student interested in deep learning and neural networks

Follow

Towards Data Science

Sharing concepts, ideas, and codes.

Following

More from Towards Data Science

12 Things I Learned During My First Year as a Machine Learning...

Daniel Bourke
Jul 6 · 11 min read ★ 2.6K

More from Towards Data Science

How a simple mix of object-oriented programming can sharpe...

Tirthajyoti Sarkar
Jul 5 · 11 min read ★ 1.4K

More from Towards Data Science

What Separates Good from Great Data Scientists?

Amadeus Magrabi
Jun 30 · 6 min read 1.3K

Responses

Be the first to write a response…