Basics of Neural Network Programming

Logistic Regression Gradient descent

Logistic Regression Gradient Descent

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

English

Help us translate!

0:00

Welcome back. In this video, we'll talk about how to compute derivatives for you to implement gradient descent for logistic regression. The key takeaways will be what you need to implement. That is, the key equations you need in order to implement gradient descent for logistic regression. In this video, I want to do this computation using the computation graph. I have to admit, using the computation graph is a little bit of an overkill for deriving gradient descent for logistic regression, but I want to start explaining things this way to get you familiar with these ideas so that, hopefully, it will make a bit more sense when we talk about full-fledged neural networks. To that, let's dive into gradient descent for logistic regression. To recap, we had set up logistic regression as follows, your predictions, Y_hat, is defined as follows, where z is that. If we focus on just one example for now, then the loss, or respect to that one example, is defined as follows, where A is the output of logistic regression, and Y is the ground truth label. Let's write this out as a computation graph and for this example, let's say we have only two features, X1 and X2. In order to compute Z, we'll need to input W1, W2, and B, in addition to the feature values X1, X2. These things, in a computational graph, get used to compute Z, which is W1, X1 + W2 X2 + B, rectangular box around that. Then, we compute Y_hat, or A = Sigma_of_Z, that's the next step in the computation graph, and then, finally, we compute L, AY, and I won't copy the formula again. In logistic regression, what we want to do is to modify the parameters, W and B, in order to reduce this loss. We've described the four propagation steps of how you actually compute the loss on a single training example, now let's talk about how you can go backwards to compute the derivatives. Here's a cleaned-up version of the diagram. Because what we want to do is compute derivatives with respect to this loss, the first thing we want to do when going backwards is to compute the derivative of this loss with respect to, the script over there, with respect to this variable A. So, in the code, you just use DA to denote this variable. It turns out that if you are familiar with calculus, you could show that this ends up being -Y_over_A+1-Y_over_1-A. And the way you do that is you take the formula for the loss and, if you're familiar with calculus, you can compute the derivative with respect to the variable, lowercase A, and you get this formula. But if you're not familiar with calculus, don't worry about it. We'll provide the derivative form, what else you need, throughout this course. If you are an expert in calculus, I encourage you to look up the formula for the loss from their previous slide and try taking derivative with respect to A using calculus, but if you don't know enough calculus to do that, don't worry about it. Now, having computed this quantity of DA and the derivative or your final alpha variable with respect to A, you can then go backwards. It turns out that you can show DZ which, this is the part called variable name, this is going to be the derivative of the loss, versus back to Z, or for L, you could really write the loss including A and Y explicitly as parameters or not, right? Either type of notation is equally acceptable. We can show that this is equal to A-Y. Just a couple of comments only for those of you experts in calculus, if you're not expert in calculus, don't worry about it. But it turns out that this, DL DZ, this can be expressed as DL_DA_times_DA_DZ, and it turns out that DA DZ, this turns out to be A_times_1-A, and DL DA we have previously worked out over here, if you take these two quantities, DL DA, which is this term, together with DA DZ, which is this term, and just take these two things and multiply them. You can show that the equation simplifies to A-Y. That's how you derive it, and that this is really the chain rule that have briefly eluded to the form. Feel free to go through that calculation yourself if you are knowledgeable in calculus, but if you aren't, all you need to know is that you can compute DZ as A-Y and we've already done that calculus for you. Then, the final step in that computation is to go back to compute how much you need to change W and B. In particular, you can show that the derivative with respect to W1 and in quotes, call this DW1, that this is equal to X1_times_DZ. Then, similarly, DW2, which is how much you want to change W2, is X2_times_DZ and B, excuse me, DB is equal to DZ. If you want to do gradient descent with respect to just this one example, what you would do is the following; you would use this formula to compute DZ, and then use these formulas to compute DW1, DW2, and DB, and then you perform these updates. W1 gets updated as W1 minus, learning rate alpha, times DW1. W2 gets updated similarly, and B gets set as B minus the learning rate times DB. And so, this will be one step of grade with respect to a single example. You see in how to compute derivatives and implement gradient descent for logistic regression with respect to a single training example. But training logistic regression model, you have not just one training example given training sets of M training examples. In the next video, let's see how you can take these ideas and apply them to learning, not just from one example, but from an entire training set.

Downloads

Lecture Video mp4

Subtitles (English)	WebVTT
Transcript (English)	txt
Lecture slides	pptx

Would you like to help us translate the transcript and subtitles into additional languages?