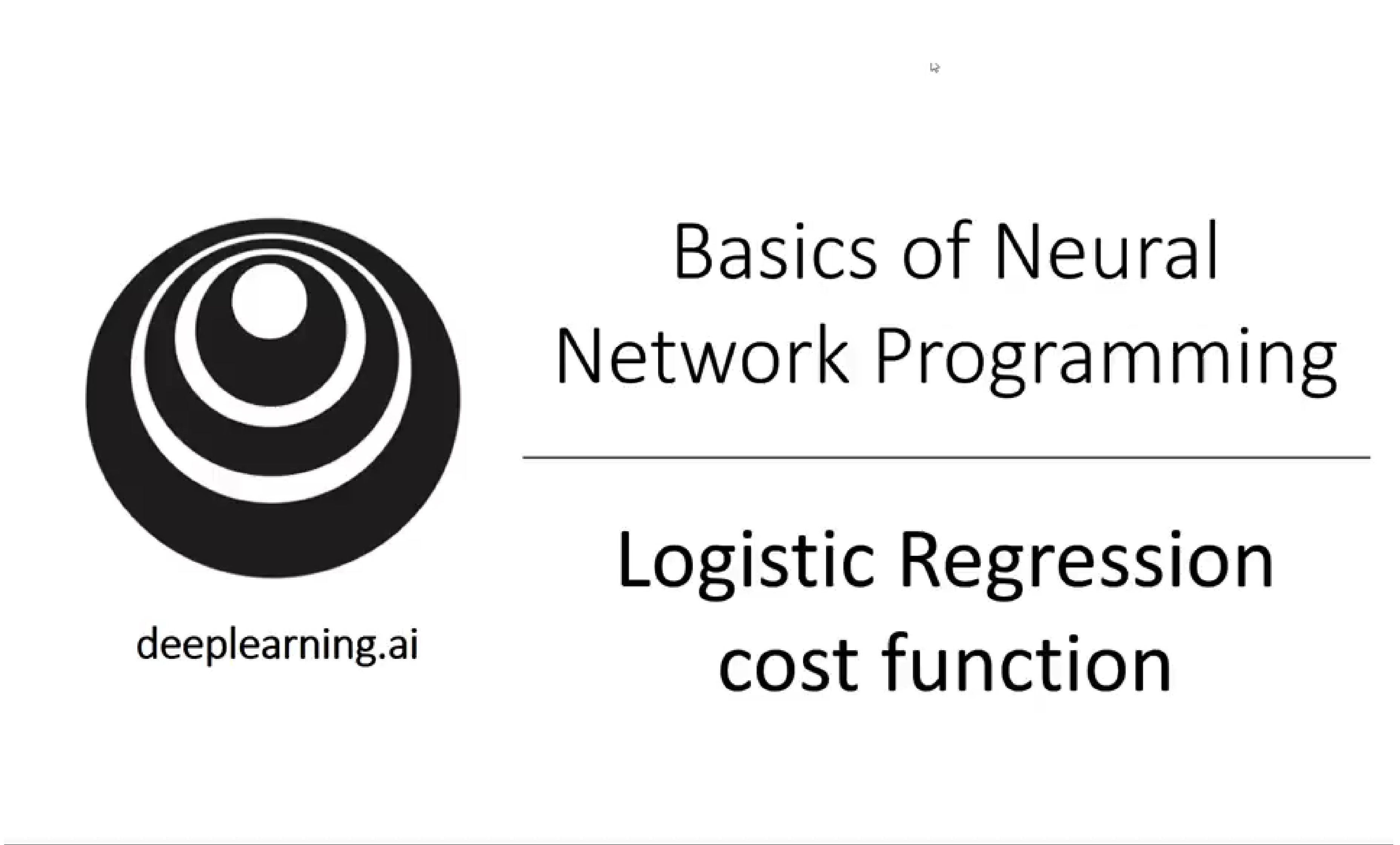

Basics of Neural Network Programming

Logistic Regression cost function

deeplearning.ai

Logistic Regression Cost Function

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

Interactive Transcript

English

Help us translate!

0:00
In a previous video, you saw the logistic regression model. To train the parameters W and B of the logistic regression model, you need to define a cost function. Let's take a look at the cost function you can use to train logistic regression. To recap, this is what we had to find from the previous slide. So your output y-hat is sigmoid of w transpose x plus b where a sigmoid of Z is as defined here. So to learn parameters for your model you're given a training set of m training examples and it seems natural that you want to find parameters W and B so that at least on the training set, the outputs you have. The predictions you have on the training set, which we only write as y-hat (i) that that will be close to the ground truth labels y_i that you got in the training set. So to throw in a little bit more detail for the equation on top, we had said that y-hat is as defined at the top for a training example x and of course for each training example, we're using these superscripts with round brackets with parentheses to index and to differentiate examples. Your prediction on training sample (i) which is y-hat (i) is going to be obtained by taking the sigmoid function and applying it to W transpose X, (i) the input that the training example plus V and you can also define Z (i) as follows. Z (i) is equal to the W transpose x (i) plus b. So throughout this course, we're going to use this notational convention, that the superscript parentheses i refers to data. X or Y or Z or something else associated with the i-th training example, associated with the i-th example. That's what the superscript i in parentheses means. Now, let's see what loss function or error function we can use to measure how well our algorithm is doing. One thing you could do is define the loss when your algorithm outputs y-hat and the true label as Y to be maybe the square error or one half a square error. It turns out that you could do this, but in logistic regression people don't usually do this because when you come to learn the parameters, you find that the optimization problem which we talk about later becomes non-convex. So you end up with optimization problem with multiple local optima. So gradient descent may not find the global optimum. If you didn't understand the last couple of comments. Don't worry about it, we'll get to it in later video. But the intuition to take away is that this function L called the loss function is a function you'll need to define to measure how good our output y-hat is when the

true label is y. As square error seems like it might be a reasonable choice except that it makes gradient descent not work well. So in logistic regression, we will actually define a different loss function that plays a similar role as squared error, that will give us an optimization problem that is convex and so we'll see in that later video becomes much easier to optimize. So, what we use in logistic regression is actually the following loss function which I'm just like right up here, is negative y log y-hat plus one line has y log, one line is y-hat. Here's some intuition for why this loss function makes sense. Keep in mind that if we're using squared error then you want the squared error to be as small as possible. And with this logistic regression loss function, we'll also want this to be as small as possible. To understand why this makes sense, let's look at the two cases. In the first case, let's say Y is equal to one then the loss function y-hat comma y is justice for us to write this negative sign. So this negative log y-hat. If y is equal to one. Because if y equals one then the second term one minus Y is equal to zero. So this says if y equals one you want negative log y-hat to be as big as possible. So that means you want log y-hat to be large, to be as big as possible and that means you want y-hat to be large. But because y-hat is you know, the sigmoid function, it can never be bigger than one. So this is saying that if y is equal to one you, want y-hat to be as big as possible. But it can't ever be bigger than one so saying you want y-hat to be close to one as well. The other case is if y equals zero. If y equals zero then this first term in the loss function is equal to zero because y zero and then the second term defines the loss function. So the loss becomes negative log one minus y-hat. And so if in your learning procedure you try to make the loss function small, what this means is that you want log one minus y-hat to be large. And because it's a negative sign there and then through a similar piece of reason you can conclude that this loss function is trying to make y-hat as small as possible. And again because y-hat has to be between zero and one. This is saying that if y is equal to zero then your loss function will push the parameters to make y-hat as close to zero as possible. Now, there are a lot of functions with Rafidah's effect that if y is equal to one we try to make y-hat large and if Y is equal to zero we try to make y-hat small. We just gave here in green a somewhat informal justification for this loss function will provide an optional video later to give a more formal justification for why in logistic regression we like to use the loss function with this particular form. Finally, the loss function was defined with respect to a single training example. It measures how well you're doing on a single training example. I'm now going to define something called the cost function, which measures how well you're doing an entire training set. So the cost function J which is applied to your parameters W and B is going to be the average with one of the m of the sum of the loss function applied to each of the training examples and turn. While here y-hat is of course the prediction output by your logistic regression algorithm using you know, a particular set of parameters W and B. And so just to expand this out, this is equal to negative one over m sum from i equals one through m of the definition of the loss function. So this is y (i) Log y-hat (i) plus one line is y (i) log one line is y-hat (i). I guess I could put square brackets here. So the minus sign is outside everything else. So the terminology I'm going to use is that the loss function is applied to just a single training example like so. And the cost function is the cost of your parameters. So in training your logistic regression model, we're going to try to find parameters W and B that minimize the overall costs of machine J written at the bottom. So, you've just seen the set up for the logistic regression algorithm, the loss function for training example and the overall cost function for the parameters of your algorithm. It turns out that logistic regression can be viewed as a very very small neural network. In the next video we'll go over that so you can start gaining intuition about what neural networks do. So that let's go onto the next video about how to view logistic regression as a very small neural network.

Downloads

Lecture Video	mp4
Subtitles (English)	WebVTT
Transcript (English)	txt
Video Notes	pdf

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