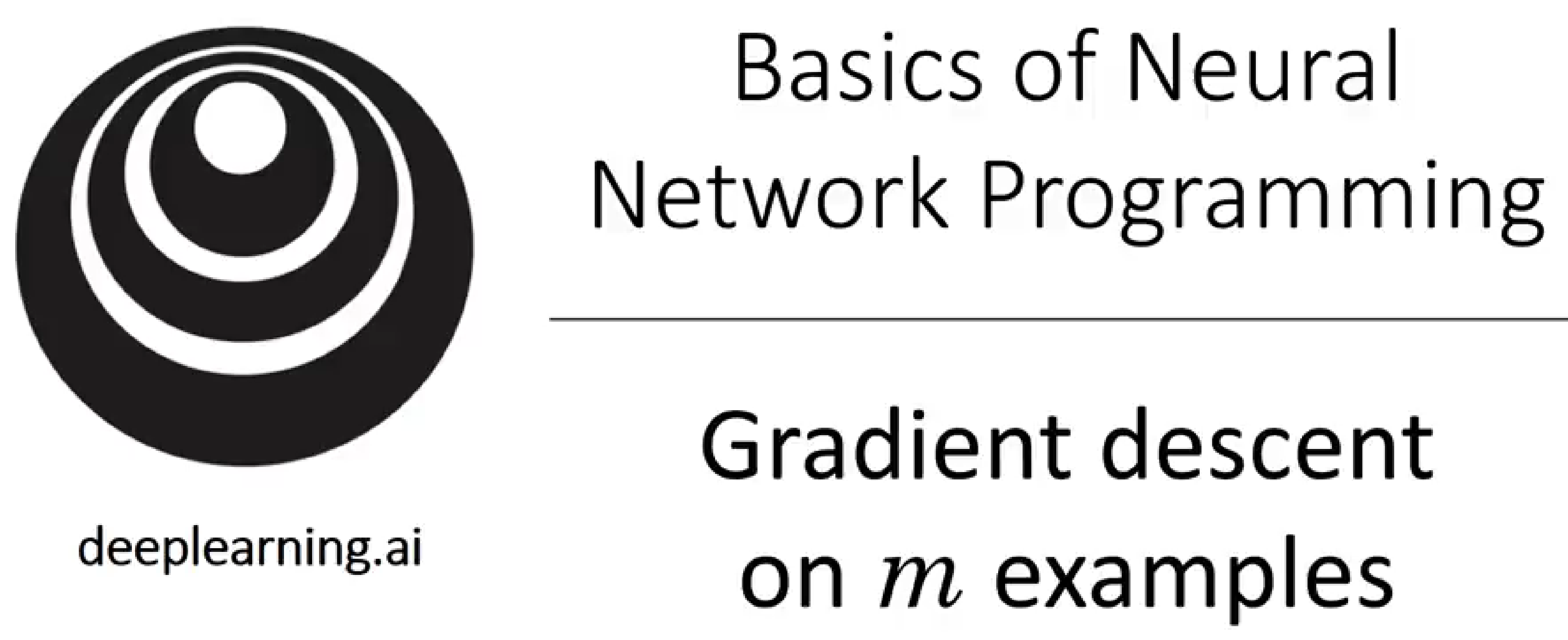
coursera

Back to Week 2 | Lessons | This Course: Neural Networks and Deep Learning | Prev | Next



deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Gradient Descent on m Examples

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

Interactive Transcript

English

Help us translate!

0:00

in a previous video you saw how to compute derivatives and implement gradient descent with respect to just one training example for religious regression now we want to do it for Emma training examples to get started let's remind ourselves that the definition of the cost function J cost function WP which you care about is this average right 1 over m sum from I equals 1 to M you know the loss when your algorithm output a I on the example why we're you know AI is the prediction on the I've trained example which is Sigma of Z I which is equal to Sigma of W transpose X plus B ok so what we show in the previous slide is for any single training example how to compute LD derivatives when you have just one training example great so d w1 d w2 and d be with now the superscript I to denote the corresponding values you get if you are doing what we did on the previous slide but just using the one training example X I Y I I was using it missing on either as well so now you notice the overall cost functions is sum was really the average because the 1 over m term of the individual losses so it turns out that the derivative respect to say w1 of the overall cost function is also going to be the average of derivatives respect to w1 of the individual loss terms but previously we have already shown how to compute this term as say d w1 I right which we you know on the previous slide show how the computers on a single training example so what you need to do is really compute these own derivatives as we showed on the previous training example and average them and this will give you the overall gradient that you can use to implement straight into scent so I know there was a lot of details but let's take all of this up and wrap this up into a concrete algorithms and what you should implement together logistic regression with gradient descent working so just what you can do let's initialize J equals 0 on DW 1 equals 0 DW 2 equals 0 DB equals 0 and what we're going to do is use a for loop over the training set and compute the derivatives to respect each training example and then add them up all right so see as we do it for I equals 1 through m so M is the number of training examples we compute CI equals W transpose X I plus B armed the prediction AI is equal to Sigma of zi and then you know let's let's add up j j plus equals y i long a I M plus 1 minus y I log 1 minus AI and then put a negative sign in front of the whole thing and then as we saw earlier we have d zi or it is equal to AI minus y i and DW gets plus equals x1 i d zi b w2 plus equals x i2 d zi or and i'm doing this calculation assuming that you have just be two features so the n is equal to 2 otherwise you do this for d w1 z w2 TW 3 and so on and GB plus equals V V I and I guess that's the end of the for loop and then finally having done this for all M training examples you will still need to divide by M because we're computing averages so d w1 if I equals m DW to divide calls m DB device equals M in all the complete averages and so with all of these calculations you've just computed the derivative of the cost function J with respect to e three parameters W 1 W 2 and B so the comment details what we're doing we're using DW 1 + DW and DP - as accumulators right so that after this computation you know DW 1 is equal to the derivative of your overall cost function with respect to W 1 and similarly for DW 2 and DV so notice that DW 1 + DW to do not have a superscript I because we're using them in this code as accumulators to sum over the entire training set whereas in contrast bzi here this was on P Z with respect to just one single training example that is why that has a superscript I to refer to the one training example either that's computer on and so having finished all these calculations to implement one step of gradient descent you implement w1 gets updated as w1 - a learning rate times d w1 w2 gives updates w2 one is learning rate times d w2 and B gives update as B - learning rate times EB where PW 1 DW 2 + DB where you know as computed and finally J here would also be a correct value for your cost function so everything on the slide implements just one single step of gradient descent and so you have to repeat everything on this slide multiple times in order to take multiple steps of gradient descent in case these details seem too complicated again don't worry too much about it for now hopefully all this will be clearer when you go and implement this in D programming assignment but it turns out there are two weaknesses with the calculation as with as with implemental adhere which is that to implement logistic regression this way you need to write two for loops the first for loop is a small loop over the M training examples and the second for loop is a for loop over all the features over here right so in this example we just had two features so n is 2 equal to 2 and X equals 2 but if you have more features you end up writing your DW 1 DW 2 and you have similar computations for DW v and so on down to DW n so seems like you need to have a for loop over the features over all n features when you're implementing deep learning algorithms you find that having explicit for loops in your code makes your algorithm run less efficiency and so in the deep learning error would move to a bigger and bigger data sets and so being able to implement your algorithms without using explicit for loops is really important and will help you to scale to much bigger data sets so it turns out that there are set of techniques called vectorization techniques that allows you to get rid of these explicit full loops in your code I think in the pre deep learning era that's before the rise of deep learning vectorization was a nice to have you could sometimes do it to speed a vehicle and sometimes not but in the deep learning era vectorization that is getting rid of for loops like this and like this has become really important because we're more and more training on very large datasets and so you really need your code to be very efficient so in the next few videos we'll talk about vectorization and how to implement all this without using even a single full loop so of this I hope you have a sense of how to intimate logistic regression or gradient descent for logistic regression on things will be clearer when you implement the program exercise but before actually doing the program exercise let's first talk about vectorization so then you can implement this whole thing implement a single iteration of gradient descent without using any fall news

Downloads

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

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