Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Mechanical Engineering
Indian Institute of Technology - Madras

Lecture - 24
Gradient Descent for Nonlinear Inverse Problem Coding example

(Video Starts: 00:19)

Welcome back. So, in this video, we will try and see how gradient descent works for the problem that we just set up. So, if you look at this, you will see that I have given the data set here. So, we are again considering the first order system and I have given the data set which I had already shown you, and we have our initial guesses for a = 35 and b = 0.004. And we want to try to find the optimal coefficients for the nonlinear inverse problem.

And we are going to use what is known as Vanilla gradient descent, forget about this Vanilla, that means it is plain gradient descent. Let us go ahead and see what we did. So, here is just like the linear regression problem, I have just set up the data this is the t data, I have taken a transpose here in order to make it a column vector right now, this is a row vector. Similarly, on the right-hand side here I have specified theta, you will see 3.1, 11.8 etcetera.

I have specified it completely. Now, I have also changed my variables in order for notational convenience. So, I called x as t, just as I told you in the theory video, and y is equal to theta, this m is simply it counts how many there are, so that is 6, it will automatically come to 6. So now we can give some initial guesses for the parameters a and b we also guessed for α, here, number of epochs.

As I told you in the last videos with gradient descent simply is the number of iterations that we are using, right now just to set our expectations, we are going to try 5 iterations. And I have set the values for a and b as 35 and 0.004. 35 and 0.004 as I said earlier are from physical parameters, and now let us write down what the model and its gradient are. now here is one place, were coding shines over analytical expressions.

I have given capital Yh, this is the function which takes an x, a and b and returns the value of the hypothesis function at that time. So as written here, this is $a(1 - e^{-bx})$, this was exactly the model that we had specified earlier. And now I have written what dyda is, notice this is not

djda. dyda is simply the derivative of this with respect to a, which was $(1 - e^{-bx})$, and dydb, as we see here is x times a, not a times b, it is a times xe^{-bx}.

So, all these we had seen in the video just before. So, at this point, you can write down what a and b are, we can ignore this for right now, and we will run this code shortly. Now here is what happening, when we start doing gradient descent. Notice the final step, this is the most important step to notice a is a - alpha times dJda and b is b - alpha times the dJdb and we actually calculate what a and dJda and dJdb are.

I have called this term as the error term, because it is the gap between what our model predicts and whatever the value of y is so we have summation of Yh - y times dyda and Yh - y times dydb. So, this was just simply programmed in and we can experiment with various values of alpha. So, I am going to run this code and we can check what happens.

So, you can see that almost immediately it has jumped up, you can see that the value of b has now jumped up to 3000 something right from our initial guess and as this happens, this happens to be a very non-physical guess. I can try reducing the value of alpha but more importantly, let me just show you what happens step by step. If I take here and run this.

I can check the value of dJda, which is minus 5 point something and I can also check the value of dJdb which is minus 3.125 into 10 to the power 4 that is a really high value and that is what is blowing this entire thing up. so, if I continue, it will give me really bad results. So, I can see that maybe I should reduce the value of alpha. So let us say I make it 10^{-4}, 1 e^{-4} that means 10^{-4}, not e^{-4}, 10^{-4} within MATLAB.

Again, this is stuck here, but I will just continue and you will see this is a little bit smaller. But this also these values that you see here of b also happen to be really high, if I continue for let us say 500 epochs you will see it is just stuck there and these are not actually very physical values. So, this trouble tends to happen with gradient descent with nonlinear regression, so if I try other values, it still causes a lot of problems.

I will share you the details here. But what tends to happen with gradient descent is it does not quite converge to physical values with any sort of alpha for this problem. Now, there is a

solution to this, there is a solution to this, you can actually find out some ways of solving it. So, notice b was this and b is really, really high here, so the b it is predicting is really high, and that happens to be quite unphysical.

So, I am not going to argue from a heat transfer right now here, but because my aim is to actually move on to a different algorithm, which happens to work very, very physically for this problem. So, what we are going to do is actually re-examine how we were solving this nonlinear linear regression problem and come to a different algorithm called the Gauss Newton algorithm.

So, we will do that in a series of steps in the next few videos. So, I will see you in the next video. Thank you.

(Video Ends: 06:58)