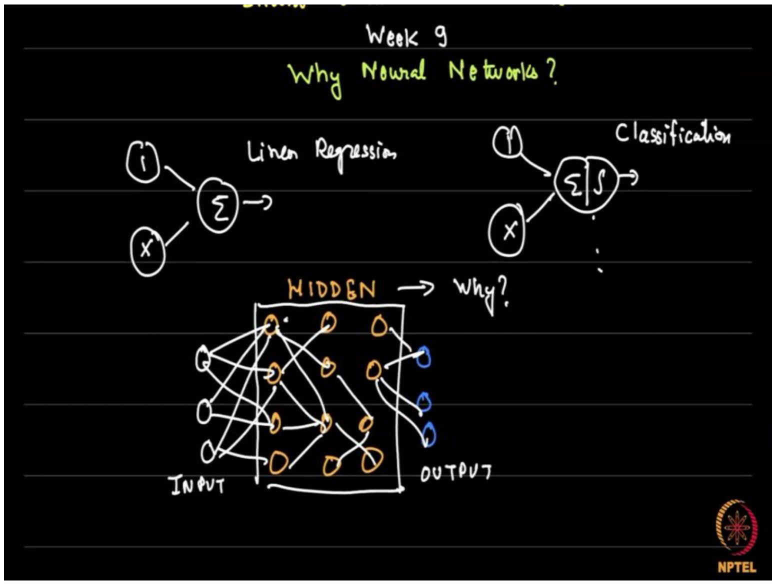
Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology, Madras

Lecture - 51
Linear Separability and Neural Networks

(Refer Slide Time: 00:18)

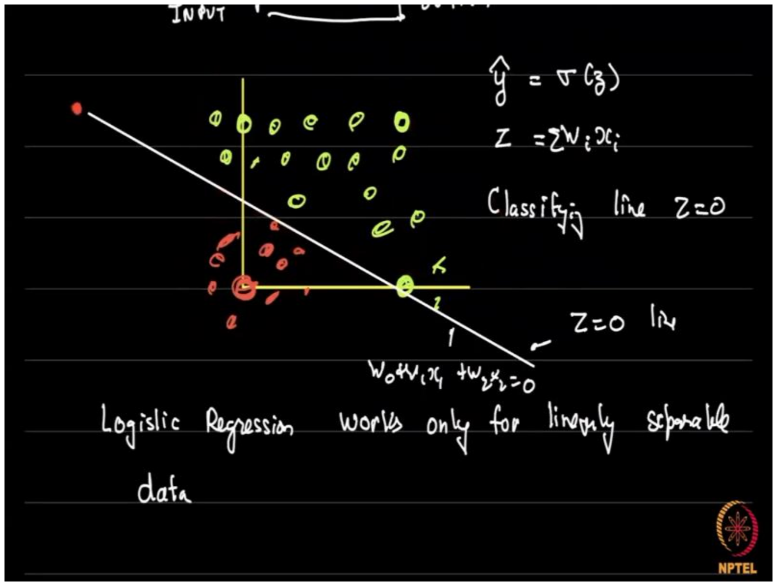


Welcome back, in this final video of this week 9 of inverse methods in heat transfer, I want to give you a brief argument for why neural networks are required. We already saw that a simple network such as this is capable of doing linear regression. so that is our regression task and a simple network like this with just a summation and a sigmoid is able to do classification. Even for multi-class classification we just need to add more neurons here.

So, why these neural networks more specifically a neural network is exactly this kind of network except, it has more layers in the middle. So, you have these intermediate calculations, so suppose you are actually doing a true multi-class problem the cat, dog, horse the problem that I talked about, the output would be after a few hidden layers. What these hidden layers are? I already explained once but I will also show it to you in greater detail in the next week's video.

But you see at the end of it is just neurons connected to each other. So, I am not drawing a fully connected layer here but you can imagine that we have these multiple connections throughout between input and output.

(Refer Slide Time: 02:05)


INPUT
$\hat{y} = \sigma(z)$
$z = \sum w_i x_i$
Classifying line z=0
z=0 line
$w_0 + w_1 x_1 + w_2 x_2 = 0$
Logistic Regression works only for linearly separable data

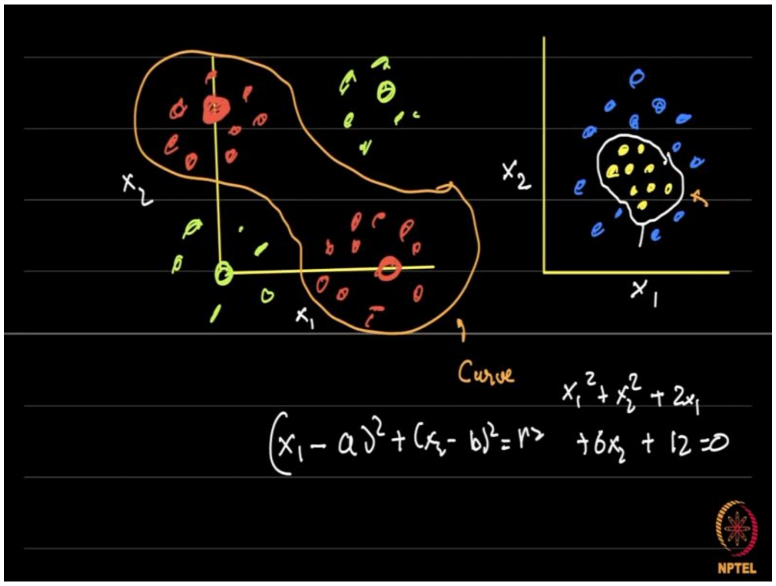
Now why these hidden layers what exactly is the need for this. So, I will briefly argue about this there is a longer argument to be made here, I can make a more precise argument but considering time and also the focus of this course which is not really on neural networks per say, but on for you to gain some intuition about new inverse methods. I am going to give you only a simple argument from the OR gate example that I showed.

So, remember the OR gate example that we had. We had this simple data set three of the points classified in one way and one of the point classified in another way. Now of course this is a simplification of actual data which might look like this. It might cluster more like this with a few unknown points here. So, this might be the actual program, what we saw was in all cases where we had $\hat{y} = \sigma(z)$ and where z was a linear combination of all the x_i the classification was via a line z equal to 0.

So, it was always a straight line which represented $w_0 + w_1x_1 + w_2x_2 = 0$. Now you might remember again if you do not, please do take me on trust you might remember from three-dimensional geometry that this actually is the equation of a line and if I put $w_3x_3 = 0$ that would be the equation of a plane in basically 3D but in all cases it is a straight line. Now how is this a problem?

The problem is that logistic regression or even multi-class classification the way we showed it works only for linearly separable data. I am going to qualify this shortly but only linearly separable. What does linearly separable mean? It means if you look at this two these two sets of points you can actually draw a line in the middle and separate them.

(Refer Slide Time: 04:33)


x_2
x_1
x_2
x_1
Curve
$(x_1 - a)^2 + (x_2 - b)^2 = r^2$
$x_1^2 + x_2^2 + 2x_1 + 6x_2 + 12 = 0$
NPTEL

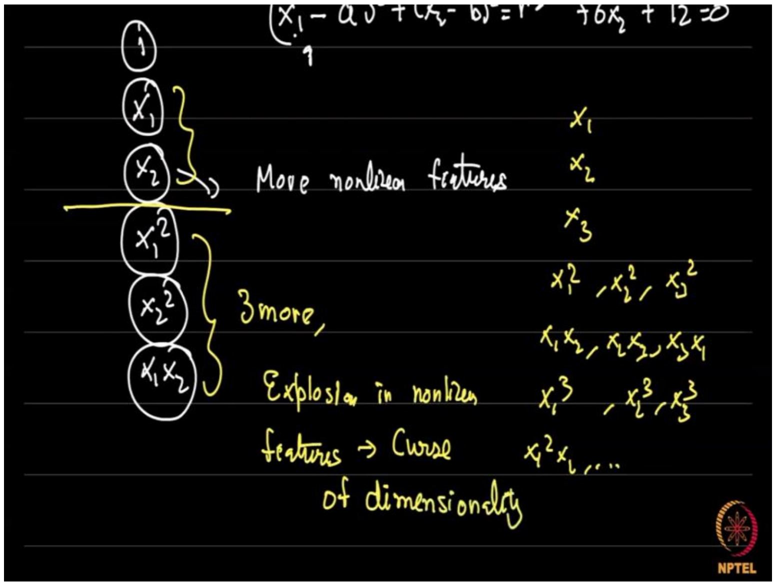
What would be the example of something that is not linearly separable? So, there is a famous historical example called the XOR gate, so you could have data of this sort. Now see there is no line if I draw a line here, I get one wrong not only do I get one wrong well occasionally by classification it is okay to get a few classifications wrong. So, you might have some data spread like this.

But this is an entirely different beast because remember this actually represents something a little bit deeper. So, if you have data of this sort, you have to get a lot of points wrong in fact you will get entire clusters are wrong, if you just draw any separating line at all. So, if I draw a separating line this way it would be wrong this way all of these would be equally wrong. Let me show another data set which is not linearly separable.

And that could be some data set like this, some set of points here and another set of points outside. Now you might say that it is not a line but at least a curved line will work maybe a circle. Let us say this is the variable x_1 and this is the variable x_2, x_1x_2. Similarly, you could think of a classifying curve like this, but the equation would no longer be z equal to 0. In fact, this one we can say something like let us say assuming this is a circle obviously it is not a circle.

But let us say it is a circle you could have something like $x_1{}^2 + x_2{}^2 + 2x_1 + 6x_2 + 12 = 0$ or $(x_1\text{-}a)^2 + (x_2\text{-}b)^2 = r^2$. So, this curve would be a curve like that.

(Refer Slide Time: 06:53)

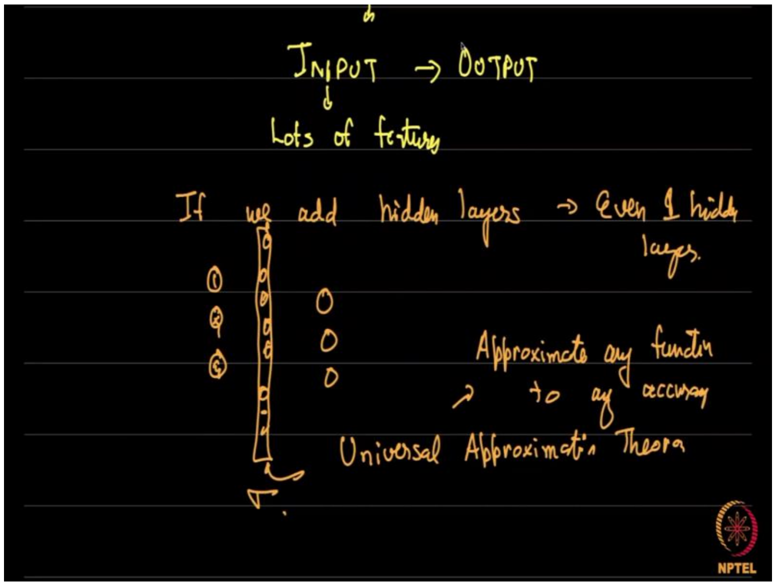


But notice this is no longer linear in x_1 and x_2 in fact, if I have to add neurons here just like we were doing with linear regression in the earlier weeks, I will explicitly have to add $x_1{}^2x_2{}^2$ something like x_1x_2. So, this leads to more non-linear features, again this might not seem like a serious problem. So, what we have decided so far is that in case you want to use pure logistic regression with just x_1x_2 it will only give lines.

Why will it give only lines? Because the line is z equal to 0 and z is always $w_1x_1 + w_2x_2$. If you want $w_1x_1{}^2$ you have to add that and if we add that we get more features. Now notice if I want a quadratic curve, I have to add three more if you just have to, now if it was $x_1x_2x_3$ you will have to add x_1 square x_2 square x_3 square x_1x_2, x_2x_3, x_3x_1. Now if I have to add cube then of course it is a whole lot more $x_1$cube x_2 cube x_3 cube x_1 square x_2 etcetera.

This leads to an explosion in non-linear features, this is called in machine learning the curse of dimensionality. The idea being that the more features you add, it is not possible for you to just get rid of it by just putting something like this simple case. What is simple about this? You had one input and direct output.

(Refer Slide Time: 09:05)


INPUT → OUTPUT
Lots of features
If we add hidden layers → Even 1 hidden layer.
Approximate any function
to any accuracy
Universal Approximation Theorem
NPTEL

So, cursive dimensionality has a problem when input directly relates to output, this will require lots of features. On the other hand, it turns out that if you add hidden layers, in fact even one hidden layer and you keep it arbitrary. So, I have let us say x_1x_2 and this 1 and I add a huge hidden layer here, we do not know what it means and then give an output, that you have access to what is known as the universal approximation theorem.

The universal approximation theorem says that all your computation load of adding lots of layers here, you can dump them here, provided there is a sigmoid or other non-linearities which I will talk about in the next week. So, provided you add one hidden layer with a sufficient number of neurons, you can actually without extra computation or without extra features here you do not need any extra features here.

Universal approximation theorem says you can approximate any function to any accuracy. Now this looks too good to be true, but it is true in that you can approximate any function to any accuracy, provided you are free to choose a lot of neurons. Now you do not explicitly have to provide x1 square x2 square which is operating on the data set you just keep on increasing this number of layers and this computation basically takes care of having approximating your function.

The disadvantage of course is that you have to add a lot of weights. But we are happy with that because that computation is a little bit easier to do rather than actually calculate arbitrary functions of our input. So, what we saw in this video is that logistic regression will do only linearly separable tasks, for more arbitrary tasks it makes sense to include you mean intermediate calculations and how to do these intermediate calculations and how to find the gradients and how a neural network works like.

This we will see in the next week and the next week you will see how we can also directly apply what all we have seen to inverse problems in heat transfer. So, I will see you in the next week, thank you.