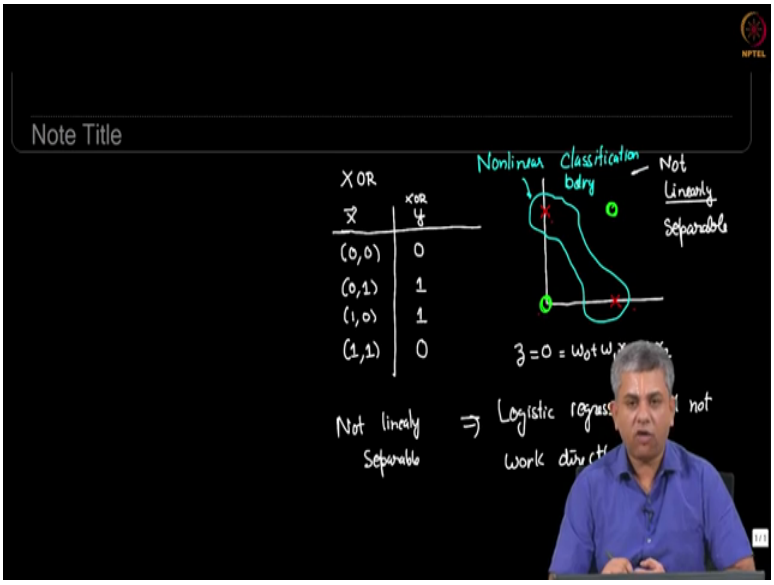
Machine Learning for Engineering and Science Applications
Professor Dr. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology, Madras
XOR Gate

In this video, we will be looking at the XOR gate. So you might recall that in the previous videos we saw that all elementary gates such as OR gate, AND gate, etcetera could be represented as simple one layer, or 2-layer neural networks basically based on simple logistic regression. So one of the first historical examples in the book, Perceptrons by Minsky et cetera, one of the first counter examples for what all neural networks could do or what all elementary perceptrons could do was the XOR gate.

(Refer Slide Time: 0:58)



So what is XOR? So XOR, we will write once again as a truth table, so XOR gives 1 or 2 as an output. So I will just write it XOR here. If x and y, or x1 and x2 are different, it gives 1 as output. If both are the same, it gives 0 as output. So once again let us try and represent this graphically in order to see what the problem is for logistic regression. So in case of 00, and 11, you get 0 and in case of 01, and 10, you get 1. So we have two classes. This red class or the x class and the o class.

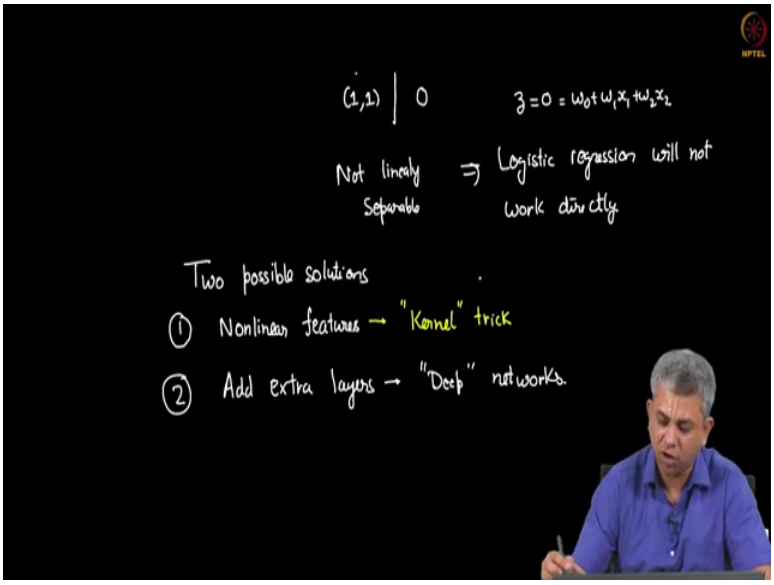
Now we know that the logistic regression the way it works is the z equal to 0 line which was the w0, plus+ w1 x1 plus+ w2 x2 line equal to 0, would function as the classifying line. But unfortunately you can see that no classification line here would work. Any line that you do

cannot actually separate the axis out from the o's. This case is called, this is not linearly separable. Please notice the word, linearly. We can of course make classification boundaries that are non-linear, for example this.

This would be a non-linear classification boundary. But linearly these two classes cannot be separated, so this is fundamentally the problem, which means since this is not linearly separable you cannot use logistic regression in order to separate these two classes. However we will see that one simple trick makes it work. Let me now talk about how this problem can be solved. So there are two possible solutions to the XOR problem. The first is to use non-linear features.

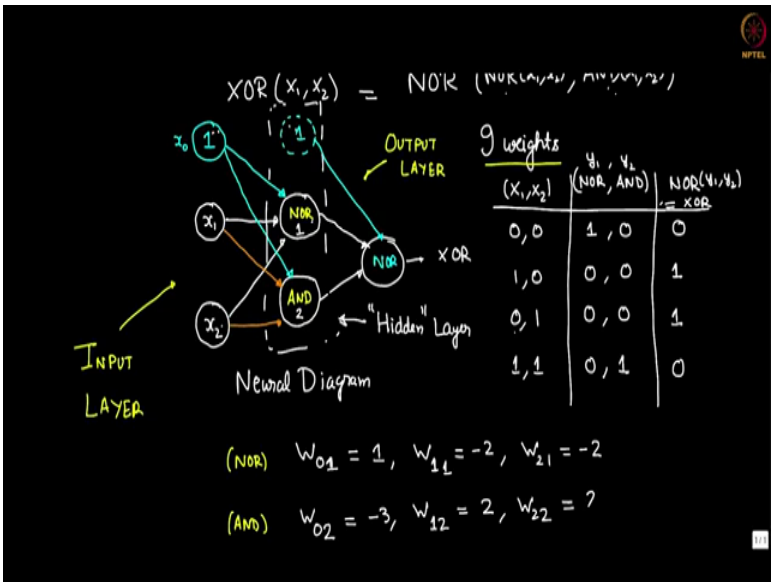
You have to be a little bit careful with how you use non-linear features. We will see later on in the course this kind of idea uses something called the kernel trick. But very simply we will have to start using other factors rather than just x1 and x2 in order to create a new boundary whose classification boundary would actually be non-linear function of x1 and x2 and not just a linear function.

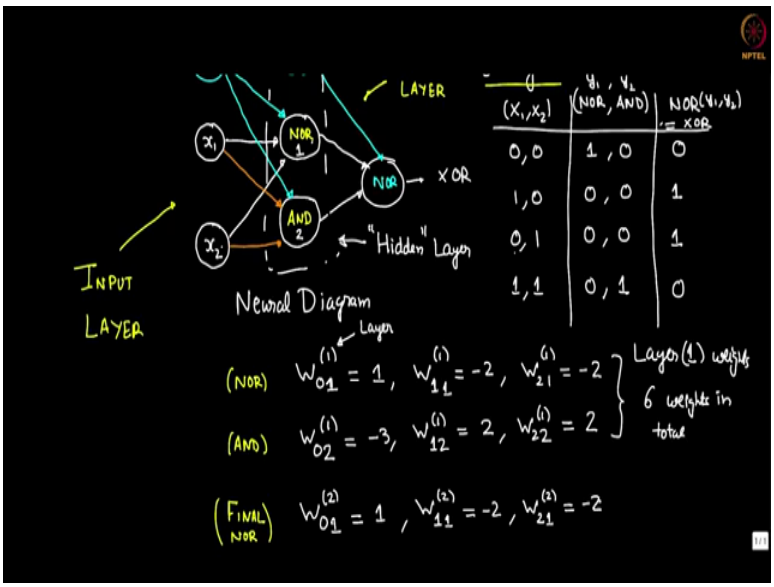
(Refer Slide Time: 4:55)



There is another possibility which is to add extra layers. I will talk shortly about what this means. This is what is called Deep networks. Technically speaking if you add more than two layers, that is when you will get a deep network. But I will just call abuse notation a little bit and call this a deep network.

(Refer Slide Time: 5:24)





Let me explain how this comes about. So we know that XOR of two inputs, x1 and x2 can be written as NOR of NOR and AND. That is XOR can be written as a combination of elementary gates that we have already looked at. Recall we already have weights for the NOR gate, for the AND gate et cetera, so which means we can in some sense write XOR in terms of those elementary gates that we have already discovered. How would this look? Suppose I have x1, x2. As usual I add a biased unit which is simply 1. We can call it x not if we wish.

So this network here or this expression here now has NOR of x1 and x2 first. So let us say I have a NOR here. Now this is a combination of these three. Let us call these weights something,

remember for the NOR gate w not was 1, w1 was minus- 2, w2 was minus- 2. I am going to erase this here for a particular reason, otherwise the figure would get very clattered. Now I also have an AND gate, let us call this AND here. This is another linear combination of x1 and x2. So we need some notation which we will use shortly. But remember that these weights, these new weights are actually different.

If you recall, the AND gates where we had used minus- 3, 2 and 2. These were different from the weights that we used for NOR gate. Let me erase these two, we will use some notation shortly which will make this whole thing clear. Now after these the output of these NOR and AND goes to another NOR. So you have yet another NOR gate which needs yet another biased unit. These weights once again were the same weights as this NOR gate.

So you can now see you have 1, 2, 3, 4, 5, 6 weights here and three more weights here. We have a total of 9 weights. The output of this as we know is XOR. I will show this to you quickly in a truth table so that you can convince yourself. X1, x2, 00, 10, 01, 11, I will just write both NOR and AND as output here. So NOR of 00 is 1, NOR of 10 is 0, this is 0, this is 0. And the second one AND of 00 is 0, of 10 is 0, 01 is 0 and 11 is 1. Now if I do NOR again of this output, NOR of 10 is 0, 1, 1, 0 which is equal to the XOR here.

So let us call this something just to be clear, y1, y2. This one is NOR of y1, y2. So this is the XOR gate which you get as this combination and we have now written what I will call the neural diagram. The difference of this diagram from the previous diagrams is it has an intermediate layer here which is the new invention in some sense. This intermediate layer is called the hidden layer, that is because it is neither the input nor the output layer which is all we had been working on before. This layer here is often called the input layer and this layer here is called output layer.

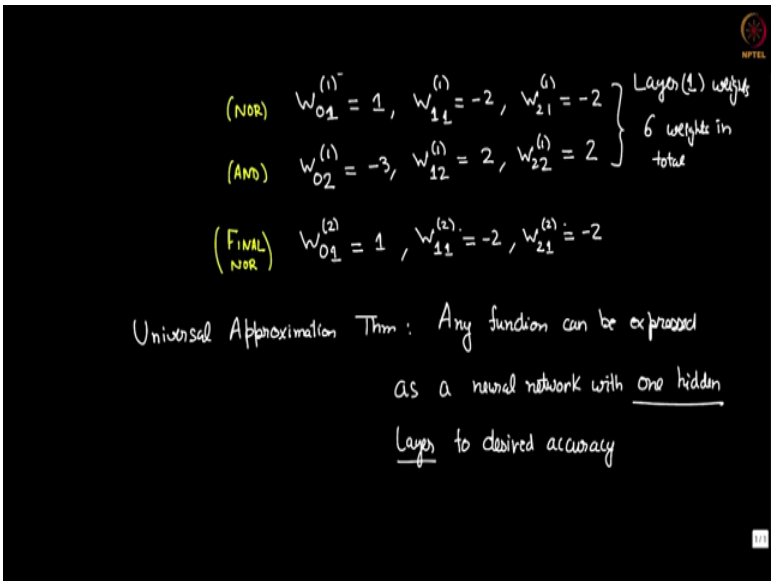
Let us use some simple terminology. Because now we have 9 weights, we need to give some terms, we can simply call them w0, w1, w2 et cetera. So let us call this as unit 1, this as unit 2, I had called it y1 and y2. We will make this a little bit more precise later on. So the weight going from the first unit here or the zeroth unit here to this, I will call that as w 01, w 01 is from the biased unit of the input layer to the first unit of this layer. Remember there nothing that goes from here to here, there is only from here to the first unit.

So w 01 was the NOR weight which is 1. Similarly w 11 which would be this weight was minus- 2. W 21 which was from here to here was also minus- 2. These were the weights of the NOR gate. Similarly we have these weights which were the weights of the AND gate. This was from NOR, those are w 0, now this is the second unit 2. This is minus- 3. W 12, which is 2. Okay.

Now we have these weights too which we have to represent which are again weights of the NOR and we need some notation for that. Now you notice that we have run out of 0s, 1s and 2s. So we have to give some slightly better notation. So what we will do is we will call this layer as layer 1. So layer 1 is the layer that connects the hidden layer to the input layer. So these are layer 1 weights and there are six weights in total. So what would be labeled or final layer weights as, so once again this is w 0, there is only 1 unit here 1. But this is now layer 2; 1.

So these are the weights of the second layer indicated here as a superscript. You can see therefore that an XOR gate can be written as a neural network, we will call this kind of diagram a neural network, each of these sitting here are neurons. So it can be written as a neural network with one hidden layer. It turns out.

(Refer Slide Time: 14:46)

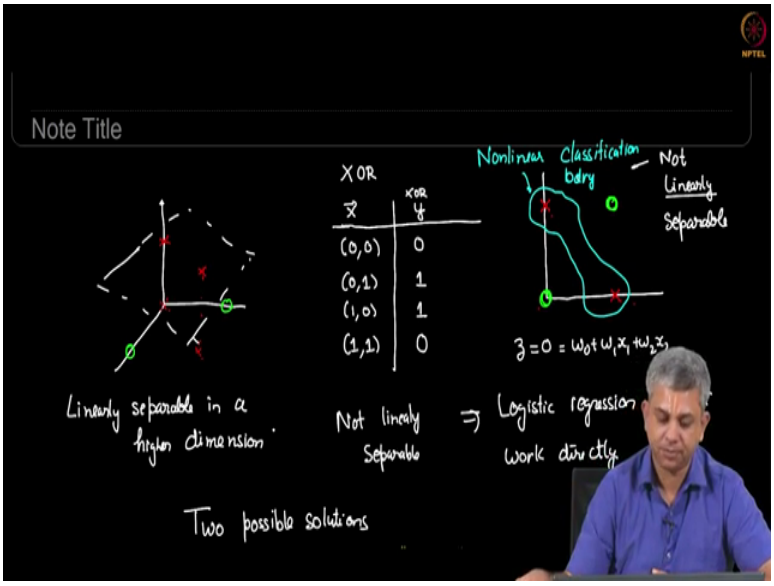


So there is a theorem called the universal approximation theorem which I am going to define very loosely here that any function can be expressed as a neural network with one hidden layer. So as it turns out this one hidden layer has tremendous expressive power. So if you pre-specify your accuracy, if you have functional relationship that you would like to express and you want to

say that to the accuracy of 10 power minus- 6 I want this function to be approximated, it turns out that there exists mathematical language, there exists one neural network which would do that with a simple one hidden layer.

What is not given is how many neurons will you need in this hidden layer. As it turned out for XOR, we only needed two, we only needed two hidden neurons. It might happen that for 10 to power minus- 6 accuracy, you might need billions of neurons for some particular function. But what we do know is that it is always possible. And this is what makes deep networks very very powerful tools. Now what would be a good approximation? What number of neurons would you need et cetera, et cetera, that is always questionable. Now I would like to come with, come up, finally end with one simple interpretation of what is happening here. Let us go back to our original diagram.

(Refer Slide Time: 16:40)



We had these four points. When they were on a single plane, when they were on simply the xy plane, it was not possible to separate them using a single line. As it turns out, one can interpret though not in the particular case that we have taken one can interpret the addition of extra hidden layers as if you are creating new dimensions in the problem. So I would like you to imagine a specific case. Remember we had these four points, xx, 00, now just imagine the case where you introduced a new dimension into the problem and the two axes were sitting here somewhere above.

That the old axis that we had in the problem were actually projections of these axes that are at a plane little bit above. Now you can imagine that if I have a three-dimensional problem, I could introduce a plane somewhere in the middle, that would separate these axes out from the o's. So this will require a little bit of imagination on your part, imagine these four points are there and you lift these two axes up. Now what was not linearly separable in a lower dimensional space can actually be linearly separable in a higher dimension.

I am not going to prove this or even show this for the XOR gate but you can see this at least geometrically. As it turns out, it is useful to think about every additional layer as sometimes introducing new dimensions into the problem which are making problems that looked not separable at least linearly before, separable later in a higher dimension. In future videos we will see how to extend this idea of a simple hidden layer into a larger network. The XOR gate and the elementary gates represent good cases for you to build your intuition on how neural networks work. So I would recommend that you go over these examples a few times in order to understand what is actually happening and why a hidden layer is hidden.