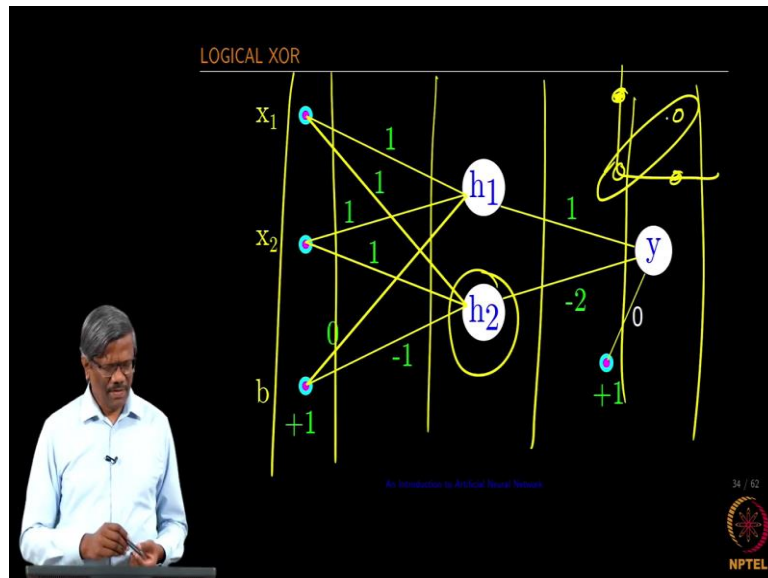


Applied Natural Language Processing
Prof. Rameshan Ramachandran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 35
Logical XOR

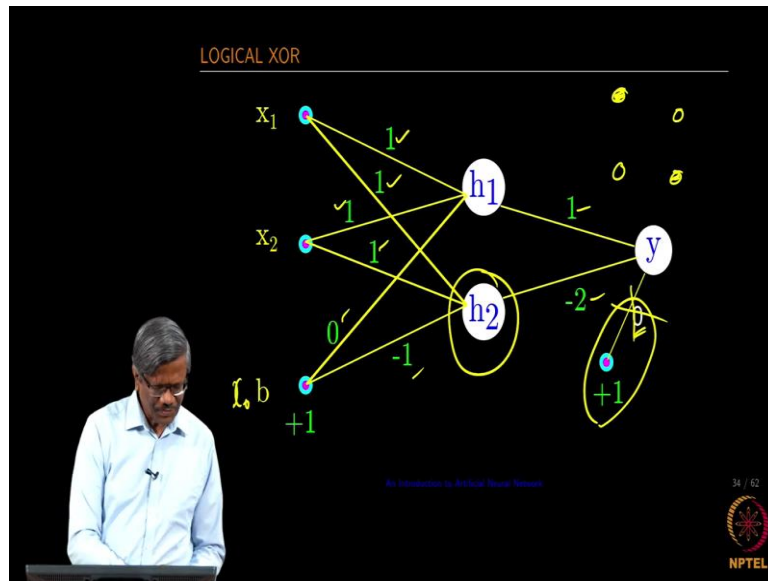
(Refer Slide Time: 00:15)



So, how do I solve the logical XOR problem? We know, I think we had seen earlier that it is not possible to solve the logical XOR problem using a perceptron. It is not linearly separable because the values are here and here, right. So, we need to be able to figure out a way to solve the problems where the boundaries cannot be a straight line and you cannot separate the class by just a straight line. So, what is the way forward?

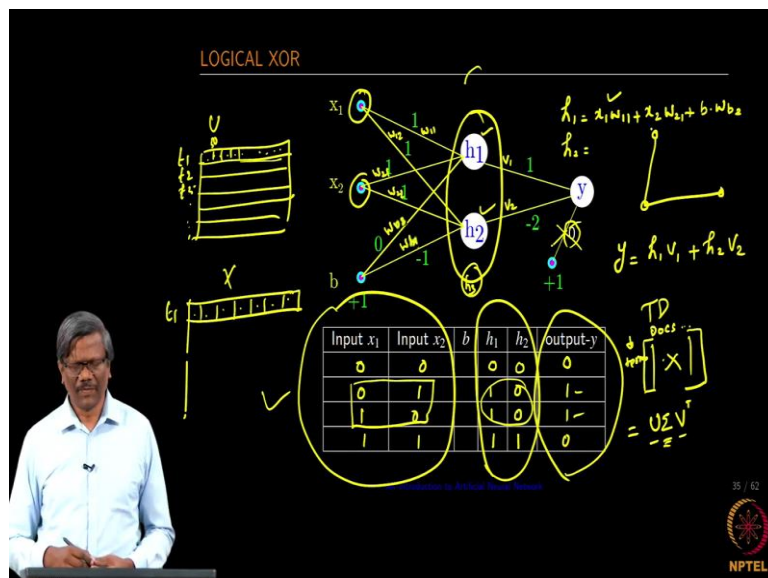
So, in this case, and you add one more neuron, ok. Now, it becomes a hidden layer. We have the input layer, we have a hidden layer and we have an output layer, ok. So, in this case again there is only one, even though it is a binary class in the XOR, single perceptron cannot solve it. So, we try to add one more neuron and see whether it is possible to solve this, ok. So, let us use this one. Let me clear all this.

(Refer Slide Time: 01:40)



Let us use this network. Assume that we have already found these values of the weights, ok. In this case the bias has no value because it is always multiplying and then by 0, it is not going to have any effect. So, you can ignore this if you want, ok. In many cases instead of having a separate bias value as we had shown earlier, it will become part of the input. So, we can consider this as x naught. So, every input will be added with the bias value.

(Refer Slide Time: 02:15)



So, in this case if you use the standard computation model that we had described earlier, ok. So, here h_1 equal to $x_1 w_{11}$, ok. So, we call this as w_{11} , w_{12} , w_{21} , w_{22} . So, for h_1 the weights are the x_1 is contributing its value through w_{11} , ok. So, that is why we have it here. x_2 is contributing the value to h_1 through w_{21} and then the bias is contributing its value through b into w_{b2} , right. So, you can compute h_1 and h_2 in this fashion for every input that you are providing, ok. So, in this fashion h_1 could be computed and then y could be computed using let us call this as v_1 , v_2 , y is equal to $h_1 v_1$ plus $h_2 v_2$ and if you have some contribution from the bias if it is not equal to 0, we can add that.

So, in this case since it is 0 we are not going to be considering this. We can actually ignore, ok. So, we can fill this up. So, our output would be like this, right. So, when you compute the values of this you will see that the h_1 is translated into this form, ok. So, h_1 and h_2 if you notice are the representatives of the input parameters. And then, notice that in this case there is an input space and there is a hidden space, and this is the output that you are computing and then it is also the expected output, right.

You can notice that from the input space to the hidden space there is a change. What change that you find? The values that you have here, for $0\ 1$ and $1\ 0$ have become $1\ 0$ and $1\ 0$ in this space, ok. It is very similar to what we saw in the LSI where we use the SVD, right from the term-document matrix using your let us call this as x . So, here you have the terms and here you have the documents, right. So, this is the input space that we had given to the LSIs. When you use the SVD what happened was we have U sigma and V transpose, right. It was decomposed into three different matters, this you know.

If you look at U the number of rows in U is very same as what you had in the original matrix x , right. So, that is why we used to call the elements of U which are representatives of our terms, and then we can either truncate the U based on your sigma size, right the rank of that or we can leave this. So, you can I can have let us say about 50 elements, and it will still represent term 1, term 2, term 3 and so on, right. And if you have more precision yeah have it extended to 100, or you want to have the entire position then you want do not want to lose anything you can keep the entire a vector or the row vector, right. So, this is our embedding that we were talking about.

But if you look at the actual values from the x , right. t_1 used to represent the term a frequency or t_f , I_{df} values and so on from individual documents. Now, in this case the t_1 is translated into a different space altogether and no longer represents only the terms in documents, right. So, this part you understand, right. In the same fashion if you look at this x are the input space is the $00, 01, 10$ and 11 . So, when you move it to the hidden domain it is translated into $00, 10, 10$ and 11 . So, it is now a shrunk space, right and then those values that output one has been shrunk into one piece, ok. You see what is happening here, right.

So, what happens if I increase this to one more? So, how will this look like? Right; so, try to increase the count of the hidden neuron by 1 and then see what happens to this table, ok. You see this, right. So, now we are able to shrink the space of the input into a smaller space using h_1 and h_2 and we are still able to create a boundary in the space of hidden layer. So, there is an input space that you can talk about. Let me erase some of these to show you that part.

(Refer Slide Time: 09:17)

LOGICAL XOR

Input x_1	Input x_2	b	h_1	h_2	output- y
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	0

Handwritten equations:

$$h_1 = x_1 w_{11} + x_2 w_{21} + b \cdot w_{1b}$$

$$h_2 = x_1 w_{12} + x_2 w_{22} + b \cdot w_{2b}$$

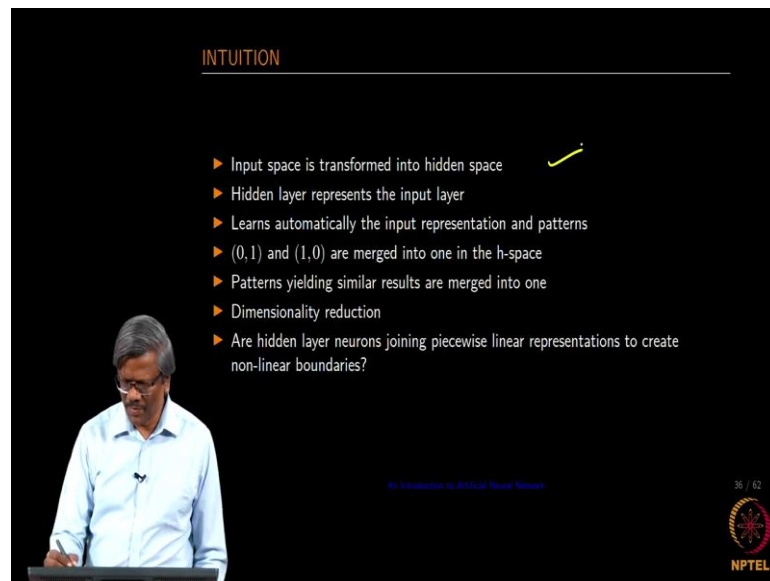
$$y = h_1 v_1 + h_2 v_2$$

So, our input spaces x_1, x_2 . We have 1 here and then this is our 0 and we have 1, right. So, this is our input space. So, in hidden space we have h_1, h_2 . So, what happened here? We have a 0 oops, right 10 is like this and then 11 is that is way, correct. So, we have one value here. So, it is easy for you to now separate these two out in this space, correct. So, there is a change from the input space to the hidden space and then the

hidden space we are able to really do the boundary, rather we are able to really compute a boundary that separates these values in a linear fashion, ok.

So, for us to get to the level where there are non-linear conditions we are going to be increasing the hidden unit and it will depend on the size of the application. For example, in many cases when you go down the lecture series you will find the hidden space would be around 100, 200 or in some cases its even 500 and so on ok.

(Refer Slide Time: 11:31)



The slide is titled "INTUITION" in orange text at the top left. Below the title is a list of seven bullet points, each preceded by a blue arrowhead. A yellow checkmark is next to the first bullet point. In the bottom left corner, there is a small video inset showing a man with glasses and a light blue shirt speaking. In the bottom right corner, there is a small red and white logo with the text "NPTEL" below it. The slide background is black.

- ▶ Input space is transformed into hidden space ✓
- ▶ Hidden layer represents the input layer
- ▶ Learns automatically the input representation and patterns
- ▶ $(0,1)$ and $(1,0)$ are merged into one in the h-space
- ▶ Patterns yielding similar results are merged into one
- ▶ Dimensionality reduction
- ▶ Are hidden layer neurons joining piecewise linear representations to create non-linear boundaries?

So, I have provided all the intuition that we have spoken about in this particular slide, ok. So, you may want to read this and then understand this cleanly. So, the last one is a question for you or try to answer this question.