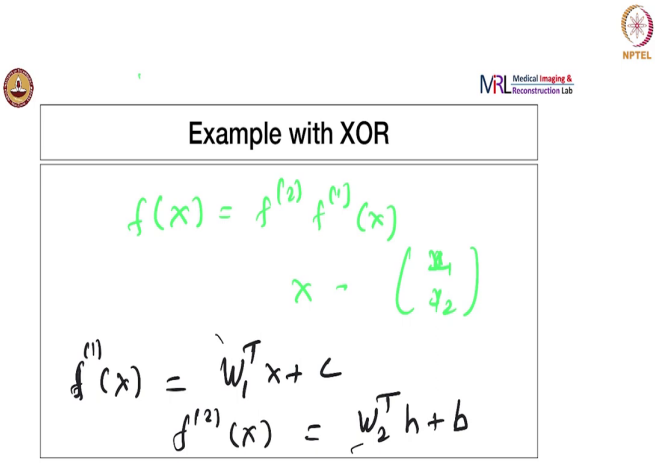
Medical Image Analysis
Professor. Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology, Madras
Lecture 42

Example with XOR

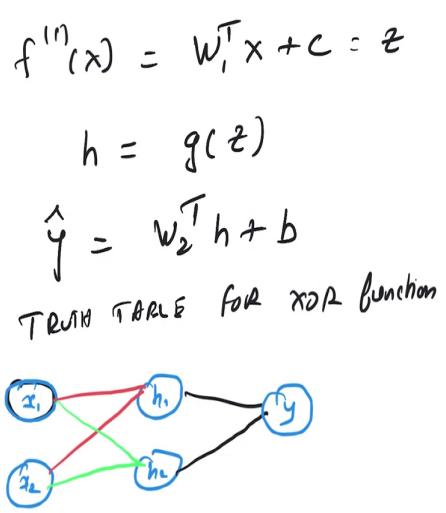
(Refer Slide Time: 00:13)



So, let us see if we can solve the XOR function using a one-layer neural network. So, our neural network, I am going to write it down as $f(x) = f^{(2)}(f^{(1)}(x))$, where this x is the inputs, x_1, x_2, two Boolean variables, x_1 and x_2. And so, for each layer, so what is $f^1(x)$, we are going to write $f^1(x) = W_1^T.X + C$, this is the input layer.

And $f^2(x)$, this is f^1, I am sorry, let me just redo this, f^1, and $f^2(x) = W_2^T.h + b$. So, this is our neural network. So, we have first, first layer weights w_1, second layer weights w_2 which maps to the output. So, we have, for a pair of inputs, we have exactly one output. So, y is the output of the XOR gate given these two binary inputs. So, in between, we are going to also have the non-linearity because, let me just go to another page, so we are going to rewrite this.

(Refer Slide Time: 02:01)

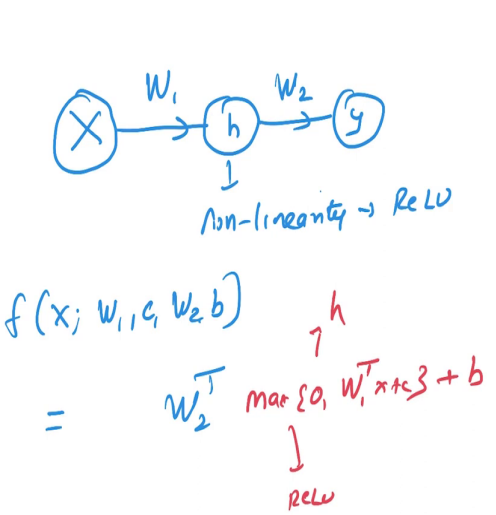


So, we have $f^1(x) = W_1^T.X + C = z$, and then we are going to this is your z and then your h is $g(z) = h$ and then your output $\hat{y} = W_2^T.h + b$. This is the general form, that we will be using for the truth table for XOR, exclusive OR function.

Now, if we do the computation this way, will it help? So, once again, we will just see, fix the weights and do this. So, for the problem that we are going to do with our neural network will look like the following. So, we have x_1, let me use different color actually, x_1, we have x_2. And we will have two hidden units.

But unlike what I showed you, when we are using the Boolean operations, and not surprising great, so here we have two hidden units. From there we have output, which is your y. This is the flow and the other way of looking at it is the following. Let us go to slide.

(Refer Slide Time: 03:52)

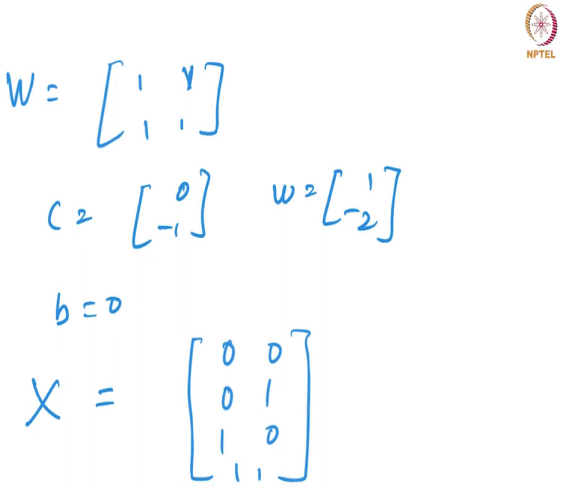


So, here we have your x as input. So, then we have your matrix w_1 to get your hidden vector h from which you have other matrix w_2 to get you to your output y, of course, implicit in that is that there is a non-linearity sitting inside. So, the non-linearity is implicit. So, this is another way of writing the same neural network. So, if we use the non-linearity as ReLU, we can do that. And if you fix the weights maybe we see what kind of results we get.

So, I leave it to you to do the computations. So, let us see, let us write our neural network down in terms of function. So, it is f(x ; w_1,c,w_2,b) now the parameters are w_1, c, w_2 and b. And we are using non-linearity which is the ReLU, which is max(0,x) we know that. So, what we have is f(x ; w_1,c,w_2,b) = W_2^T.max{ 0 , W_1^T.X + C }.

This is the hidden unit, this is h, after you apply the non-linearity, this is a non-linearity, this is the ReLU. So, this max(0 , x) is basically the ReLU plus the biased for the output and b. So, this is what, this is our neural network basically. So, we can actually write down the solution.

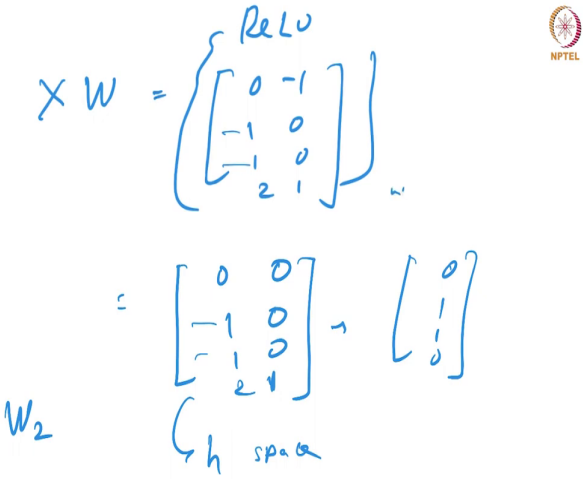
(Refer Slide Time: 06:08)



So, the computations I leave it to you that will be the questions in the homework. So, apparently, we can fix W →(1, 1, 1, 1) and c → (0 , -1) and w → (1,-2) and b → 0. So, you can, like I said you can fix the bias. So, in this case, for the same layer you have set of biases and we usually use one bias for every layer. So, now, you can walk through each one of the outputs.

So, it turns out that instead of carrying $(xw)^T.x$, you can just do the xw also. So, what is X? So, if you write that down, X is, you have the input, x_1,x_2 →(0 0 0 1 1 0 1 1), this is your X. So, you can now do and either, this is again how we write your inputs, so, you can either do w^Tx or just do xw, add the bias vector and then get the results. So, let me just write to the results.

(Refer Slide Time: 07:25)



So, then if you do that XW for the first layer, you will get XW → (0 -1 1 0 1 0 2 1), again you should do the computations. So, to finish computing the value of h, we have to now on this we apply ReLU on this. So, if you apply ReLU pointwise, which is what we discussed. So, you will get (0 0 1 0 1 0 2 1). Then, we finish by multiplying with a weight vector w_2. So, we can, again I leave it to you to apply w_2, then you can get the result as (0 1 1 0).

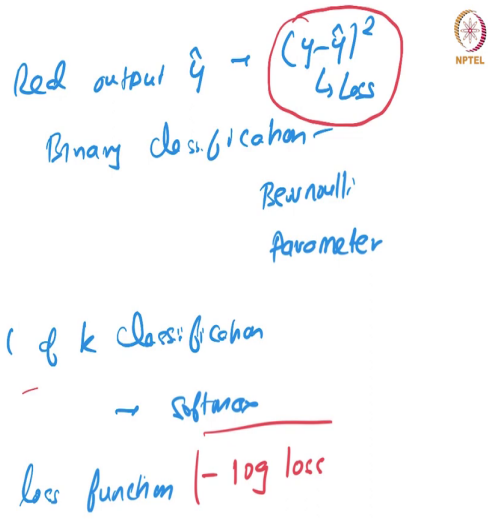
So, with this arbitrary choice of weights, you are able to get this. So, you actually do the matrix multiplication, see whether you get the w's in correct order, whether we can do $w^T x$ or xw what should be the dimensions, etc. By actually working out this by hand. So, again, this exactly what happens here is, if you look at the intermediate outputs, you will be able to see that the h, this is your h.

After you apply the ReLU, this is your h space. So, you can see that two (0 1) and (1 0) again have been mapped into one value. That is what has happened. And so, then it is easier to see, which is what I showed you when we use the Boolean operations when we split, we express the XOR gate as a function of two other gates to have an intermediate layer of Boolean calculations. And this is pretty much what happened there too.

So, in your input these two rows corresponding to (0 1) and (1 0). And if you do, once you do the xw that it got mapped into the same. After we added the bisector they got mapped to the same point once again. So, that is what helps the calculation. So, when we are going to do things like what if you do not have the non-linearity. And for instance, what if you did not do ReLu what happens, so on so forth, all that all those calculations you can do.

So, this just to walk you through the exact sequence of computations, you can also do similar computations for, you can make up a dummy data set where you can work with hand and actually do the forward problem. So, so far what we have only looked at this input multiply by weight matrix, apply non-linearity, get an output, so on and so forth till you get to the output and we looked at different kinds of output.

(Refer Slide Time: 10:32)



We looked at, so for instance, we looked output, real output $\hat{y}$, we had a binary classification where we interpreted this as a Bernoulli parameter. And then, we have 1 of k classification and this came up to be a softmax. We will do the softmax output. So, now we have these outputs. We still have to create this loss function or cost function. We looked at mean squared error when we did the linear regression problem, but for these classification problems of the loss function will be slightly different.

So, even in the case of neural network, if you are going to do regression problems, and in this case, you can still do $(y - \hat{y})^2$ as your loss. That is going to be okay. On the other hand, if you are going to be doing either binary classification or 1 of k classification, you will end up doing this log loss, negative log loss. And this we can sometimes, if you have multiple classes, you can call it binary or even for two class problems you sometimes it is referred to as also can call it binary cross entropy, some version of that is what you use.

So, we will look at the different loss functions in the next class and how they behave, and generally talk about backpropagation algorithm, which helps you update the parameters of the network. So, remember, when we looked at linear regression, we did gradient descent,

where we managed to update the parameters, the w's in the linear regression model using gradient descent.

So, similarly, for neural networks, we should be able to do it, but then slightly more complicated, because of the different layers of computation. And in that case, how do we actually calculate the delta dynamics. So, we have to update the weights in every iteration. And how do we do that, what enables us to do that is the calculating the delta w's, the back-propagation algorithm helps us do that.

And the gradient of descent is the basically the learning algorithm. So, it we can calculate the gradient of the loss function, but how we propagate it to all the weights that change in the weights, the back-propagation algorithm. So, we will get a brief look at these issues in the next week of classes, and also following that will immediately run into a convolutional neural networks for image analysis and 3d image analysis and so on so forth. Thank you.