Deep Learning

Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Neural Network - II

Hello, welcome to the NPTEL online certification course on Deep Learning. In our previous lecture we started discussions on the topic in Neural Network.

(Refer Slide Time: 00:41)

CONCEPTS COVERED

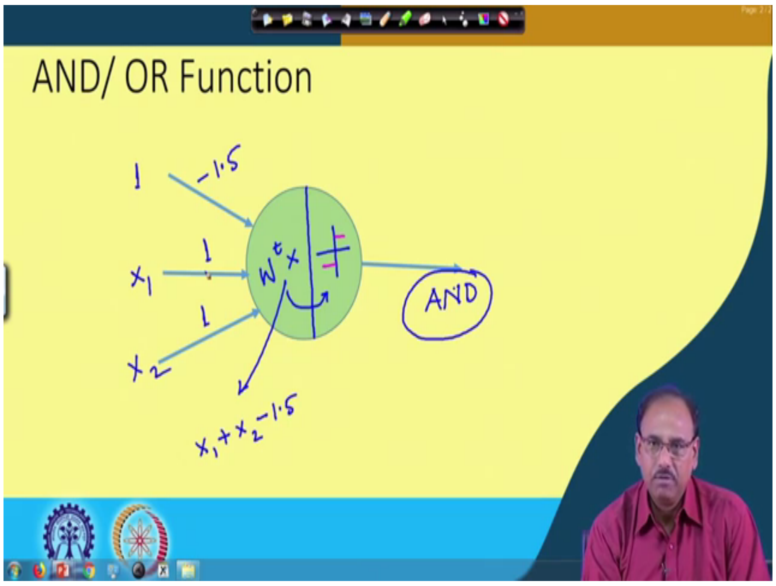
Concepts Covered:

- Neural Network
  - AND Logic
  - OR Logic
  - XOR Logic
- Feed Forward NN
- Back Propagation Learning

And we have talked about implementation of 2 of the logic functions, the AND logic and OR logic using the neural network and we have seen that AND logic and OR logic both of them being linear functions they can be implemented very easily using a single neuron. These single neurons are known as or a single layered neuron, they are known as single layer perceptron, we will see later that why they are single layer perceptron.

Today, we will talk about other implementations of neural network like XOR logic; we will also talk about feed forward neural network or a multi layer perceptron and we will also talk about how the neural networks can be trained using learning mechanism known as back propagation learning.

(Refer Slide Time: 01:47)



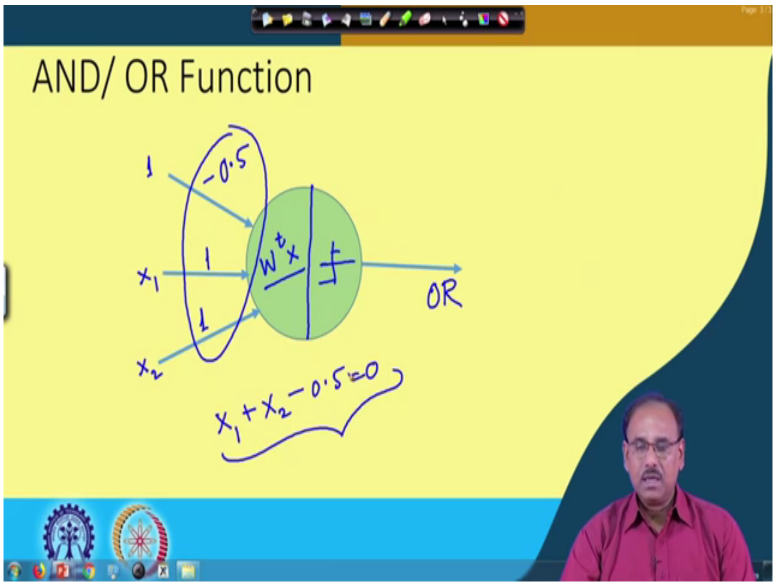
AND/ OR Function

So, let me just recapitulate what we did in our previous class, we have implemented 2 logic functions AND logic and OR logic and we said that both this AND logic and OR logic can be implemented using single neuron or single layered neural network. So, for implementation of an AND logic what we have done is we have taken our input vector to be 1 X 1 and X 2 and the weight vectors were taken as minus 1.5, 1 and 1 in the neuron I can consider that it has 2 functional parts in the first part it computes W transpose X.

So, in this case it we will compute W 0 which is nothing but minus 1.5; minus 1.5 times 1 plus 1 into X 1 plus 1 into X 2. So, effectively this function which is computed is X 1 plus X 2 minus 1.5. Now, this computed value is passed on to the second compartment of the neuron which computes the non-linearity and in this case the non-linearity is a threshold non-linearity. So, the non-linear function that we have considered is a threshold non-linearity. So, for X, W transpose, X greater than 0 the output is 1 for W transpose X less than or equal to 0 it is equal to 0.

So, as a result at the output the function that I get is an AND function. So, this is what we get in case of an AND function when the input vectors are minus 1.5, 1 and 1.

(Refer Slide Time: 03:54)



AND/ OR Function

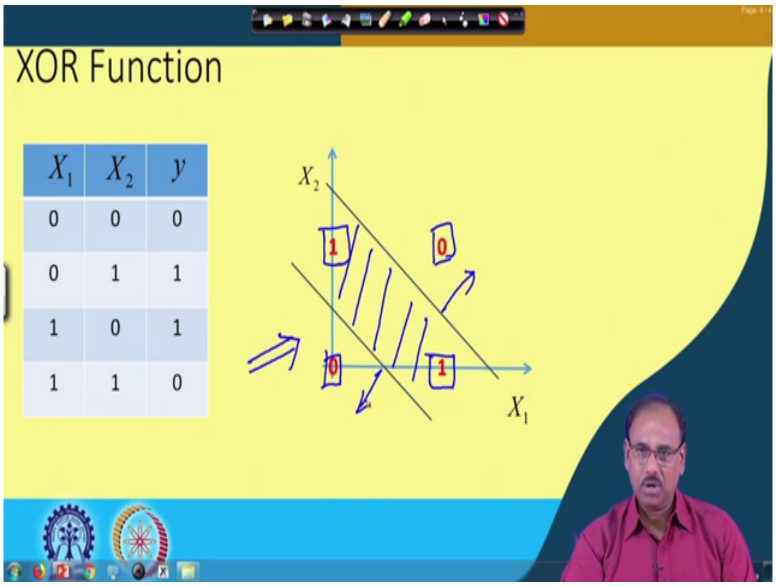
When I want to compute an OR function, I simply my input vectors remains the same that still is 1 X 1 and X 2 whereas, weight vectors are changed to minus 0.5 1 and 1 as before the first compartment computes W transpose X and the second compartment gives you the non-linearity which imposes threshold non-linearity in W transpose x. So, as a result at the output what we get an OR function or of X 1 and X 2.

And here with these weight vector this weight vector the classifier that we are doing or the separating plane that we are designing is nothing, but X 1 plus X 2 minus 0.5 that is equal to 0, that is the separating plane between the 2 classes omega 1 and omega 2. And we have also said that we could implement AND function and OR function using a simple single layered neuron or a single neuron because the classes in this case are linearly separable.

Now, what happens if the classes are not linearly separable? So, we have discussed earlier that in non-linearly separable cases or linearly non separable cases either I can have a non-linear classifier that is the boundary which is which itself is non-linear or the other approach can be that you can map the feature vectors using non-linear functions to an intermediate feature domain, where because of this non-linear mapping the in the intermediate feature domain the feature vectors will be linearly separable. And as in the intermediate feature domain the feature vectors are linearly separable.

So, I can now have linear classifiers, which classifies the features in the intermediate feature domain. So, I have 2 levels of operation in the first level of operation the feature vectors will be non-linearly mapped to an intermediate feature domain where they will be linearly separable and in the second step you these feature vectors in the intermediate feature space where they are linearly separable. In the second step I can design a linear separator or a linear classifier which classifies these vectors in the feature space. So, I have a non-linear mapping of the feature vectors in the original space and finally, a linear classifier.

(Refer Slide Time: 06:46)



XOR Function

X_1	X_2	y
0	0	0
0	1	1
1	0	1
1	1	0

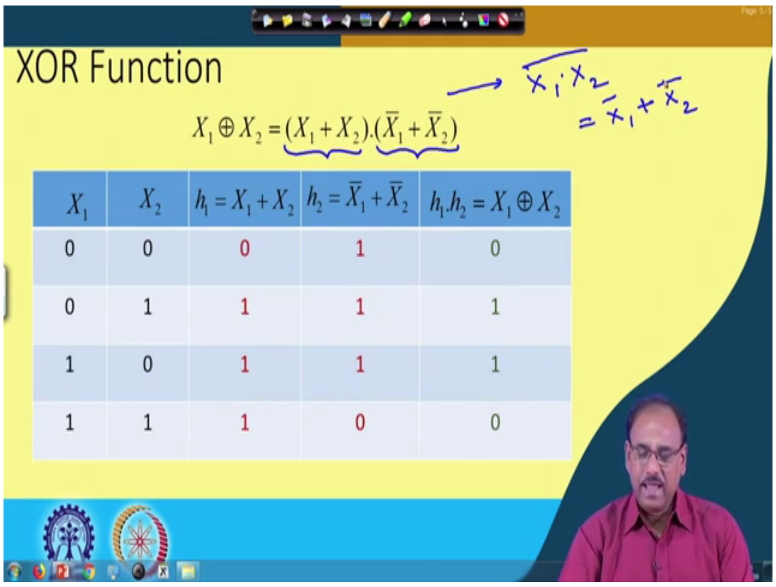
So, let us again see a very simple example, what happens that in case of and instead of AND function or OR function if I want to implement an XOR function. So, all of you know that in the in case of XOR function, when the inputs are 0 0 that is both X 1 and X 2 both of them are zeros, the output is 0. If both of them are ones that is X 1 is equal to 1 and X 2 is equal to 1, then also the output is 0, only when the X 1 and X 2 are different that is in case of 0 1 or 1 0 the output will be 1. So, that is what I have in case of XOR function.

And in the same manner as we have done before, if I want to plot this in the feature space, if I plot this feature space and plot all these feature vectors in X 1, X 2 that is 0 0 0 1 1 0 and 1 1 in this feature space, then this is the situation that I have. You find that when X 1 X 2 both of them is 0, then the output is 0 if both of them are 1 then the output is 0, if they are 0 1 or 1 0 then the output is 1.

And now find that I have a difficult situation. In the earlier cases when we talked about OR function or AND function, the classes were linearly separable on 1 side of the straight line of a straight line I had the value 0 on the other side the values are ones. But here you find that the values of zeros and ones cannot be separated by a single straight line rather I need 2 different straight lines and in between the straight lines the values are 1 and outside the straight lines the values are zeros.

So, this is a clear case where the problem is not a linear problem, but it is a non-linear problem. So, can I implement or can I implement this XOR function using neural networks or neurons?

(Refer Slide Time: 09:00)



XOR Function

$$X_1 \oplus X_2 = \underbrace{(X_1 + X_2)}.\underbrace{(\bar{X}_1 + \bar{X}_2)} \rightarrow \overline{X_1 . X_2} = \bar{X}_1 + \bar{X}_2$$

X_1	X_2	$h_1 = X_1 + X_2$	$h_2 = \bar{X}_1 + \bar{X}_2$	$h_1.h_2 = X_1 \oplus X_2$
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Let us see, so you know from your digital circuit’s course that an XOR function X 1 or X 2; X 1 XOR X 2 can be broken into a multiple stepped function. So, what I can do is, I can perform OR operation of X 1 and X 2 and I can also perform NAND operation of X 1 and X 2.

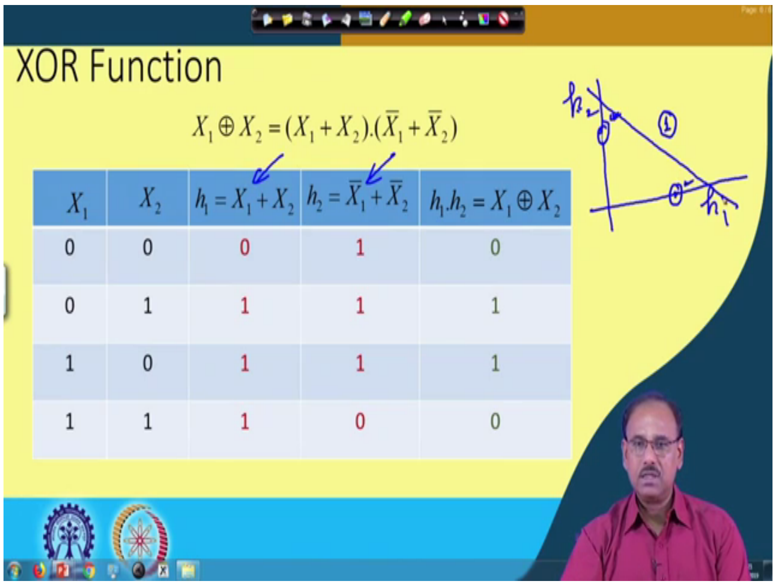
So, here what I have this first I compute X 1 or X 2 and the second one is NAND operation of X 1 and X 2; obviously, this is NAND operation because this is nothing, but

X 1 and X 2 complement of that, which is nothing but X 1 complement or X 2 complement.

So, I perform AND operation of X 1 X 2, I perform ; I perform OR operation of X 1 X 2, I perform NAND operation of X 1 X 2 and these 2 outputs the AND , OR output and AND output I end them AND them together and that gives me X 1 XOR X 2.

So, effectively what I am doing is as shown in this particular table, I am converting the input vectors X 1 X 2 to an intermediate vector h given the given by components h 1 h 2. So, here you find that if I consider that output of the OR operation is h 1 and output of NAND operation is h 2 as we have shown here.

(Refer Slide Time: 10:52)



XOR Function

$$X_1 \oplus X_2 = (X_1 + X_2).(\bar{X}_1 + \bar{X}_2)$$

X_1	X_2	$h_1 = X_1 + X_2$	$h_2 = \bar{X}_1 + \bar{X}_2$	$h_1.h_2 = X_1 \oplus X_2$
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Then when the input is 0 0 h 1 will be 0 and h 2 will be 1, when the input is 0 1 h 1 will be 1, h 2 will also be 1 when the input is 1 0 h 1 will be 1 h 2 will also be 1 and when the input is 1 1, then h 1 will be 1 and h 2 will be 0. So, as we have done before if I plot h 1 and h 2 that is the intermediate feature space. So, say this is h 1 and this is h 2 you find that when h 1 and h 2 they are having values 0 and 1 h 1 is equal to 0, h 2 is equal to 1 which is the mapped X as X 1 equal to 0 and X 2 is equal to 0.

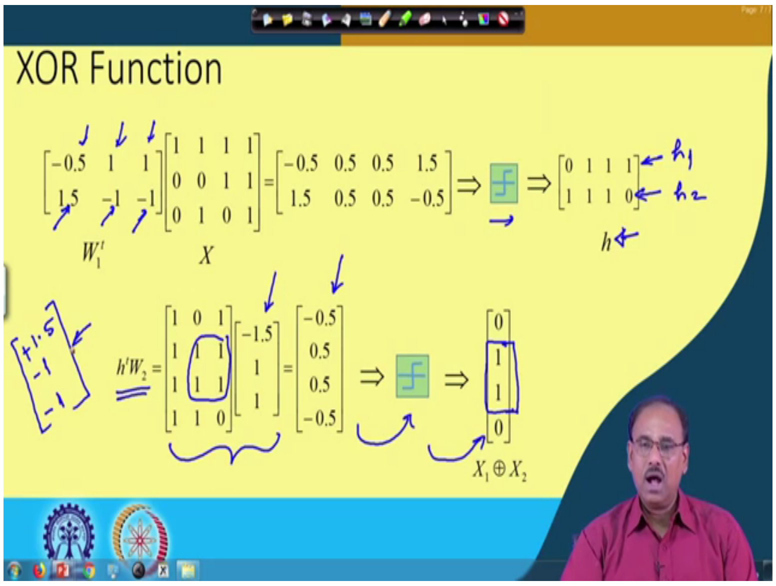
So, I have h 1 0 and h 2 1, this is where I want my XOR output to be equal to 0. Similarly, when the values are 1 1 both h 1 and h 2 they are 1 1 which is equivalent to X 1 is 0 and X 2 1 or X 2 X 1 is 1 X 2 is 0. In both these cases I want the XOR output to be

1 sorry I just made some mistake. So, over here when the h 1 h 2 both of them are 0 X 1, X 2 both of them are 0 that is h 1 is 0 and h 2 is 1; so this was correct right? So, here I want the output to be 0.

In the other case when X 1 is 0 X 2 is 1 or X 1 is 1 X 2 is 0 which are mapped to both h 1 and h 2 become becoming 1 1 I want the output to be 1. So, here I want the output to be 1 which is the case when both X 1 and X 2 are either 0 1 or 1 0 and in the other case when X 1 X 2 are 1 1 I get h 1 to be 1 and h 2 to be 0 that is somewhere over here and here my XOR output should also be 0. So, you find that now I have this is the place where I have XOR output 0, this is the place where I also have XOR output to be 0 and this is the place where I want to have XOR output to be 1.

And now we find that they are linearly separable. So, the first non-linear mapping or the first step; in the first step when I am computing X 1 or X 2 and X 1 NAND X 2, these two operations are transferring transforming my input vector to an intermediate vector given by h 1 h 2and in this intermediate space; in this intermediate feature space, the classes the problem is linearly separable. So, now, I can have a lineared classifier to classify the input vectors X 1 and X 2 by the linear classifier instead of operating on X 1 X 2, the in linear classifier operates on h 1 and h 2.

(Refer Slide Time: 14:38)



XOR Function

$$\underbrace{\begin{bmatrix} -0.5 & 1 & 1 \\ 1.5 & -1 & -1 \end{bmatrix}}_{W_1^t} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}}_{X} = \begin{bmatrix} -0.5 & 0.5 & 0.5 & 1.5 \\ 1.5 & 0.5 & 0.5 & -0.5 \end{bmatrix} \Rightarrow \boxed{} \Rightarrow \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{matrix} \leftarrow h_1 \\ \leftarrow h_2 \end{matrix}$$

$$h \leftarrow$$

$$\begin{bmatrix} +1.5 \\ -1 \\ -1 \end{bmatrix} \qquad \underline{\underline{h^t W_2}} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \Rightarrow \boxed{} \Rightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$X_1 \oplus X_2$$

So, I can represent this again in matrix form. So, what I have is in the first level I have a set of matrices given by set of weights given by the matrix W 1. So, these weights are.;so, if you look at the first 1 it is minus 0.5 1 1 which are the weight vectors we have used for an OR operation. The second one, the second weight vector is given by 1.5 minus 1 minus 1 and if you remember from our previous discussion that minus 1.5 1 1 was the weight vector corresponding to AND operation.

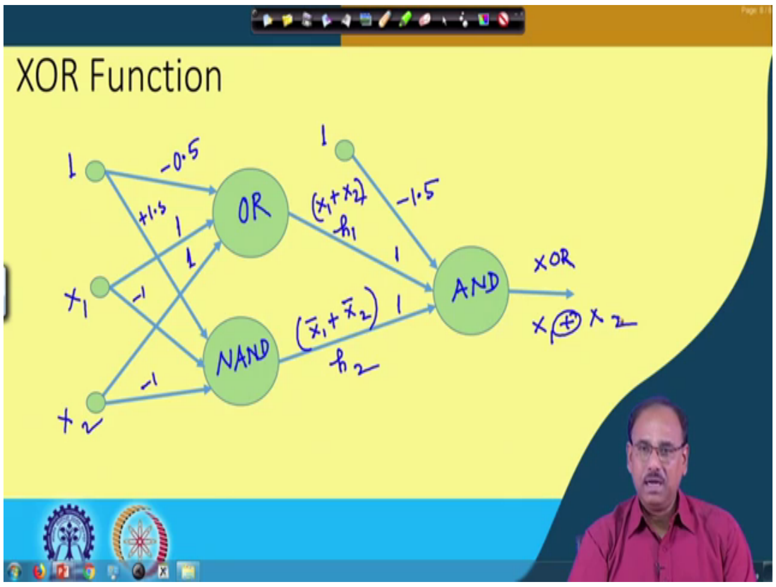
So, if I negate it, I make it plus 1 point five and this as minus 1 and this as minus 1 this is the weight vector which is used for NAND operation. So, using this weight vectors I compute W 1 transpose X, where W 1 is actually the weight matrix consisting of minus 0.5 1 one and 1.5 minus 1 minus 1. So, this is my intermediate the matrix product that I get now if I apply the non-linearity which is the threshold non-linearity I get 1 0 0 0 1 1 1 and 1 1 1 0 that is my intermediate set of vectors which is given by matrix h. So, this is my h 1 this first row corresponds to vector h 1, the second row corresponds to vectors h 2.

So, my final output if I put another weight vector, now we find that this weight vector corresponds to an AND operation that is minus 1.5, 1 and 1 this was my intermediate matrix or set of intermediate feature vectors h 1 and h 2 which is given by h. So, if I compute h transpose W 2 now, the output of this matrix multiplication becomes minus 0.5 then 0.5, 0.5 and minus 0.5 again I pass this through this threshold non-linearity and I get the output as 0 1 1 0. So, you find that when my h 1 h 2 are 0 1 or 1 0, the output becomes 0 and these are equivalent to the input vectors becoming being 0 0 or 1 1.

And when my h vector that is the intermediate features are 1 1 that is this case, the output is also 1 and this is the case when both X 1 and X 2 they are either 0 1 or 1 0. So, I have implemented this XOR function in 2 steps. So, I can consider this to be implemented using 2 layered neural networks, in the first layer I will have 2 neurons one giving an OR operation and other one giving an NAND operation and you find that because AND is a linear problem NAND is also a linear problem because NAND is nothing, but complement of AND and that is what we have done over here that plus point 1 plus 1.5 minus 1 minus 1, this weight vector actually gives a NAND operation.

So, I can implement this XOR operation which is linearly non separable, but for implementation of this as obvious from this operation that a single layered neural network or a single neuron is not sufficient rather I need three neurons and those three neurons are to be arranged in 2 layers. One neuron in the first layer will compute or operation, the other neuron in the first layer will compute NAND operation and the then the third neuron which is in the second layer will combine the outputs of these neurons in the first layer using an AND operation to give you the final output.

(Refer Slide Time: 19:12)

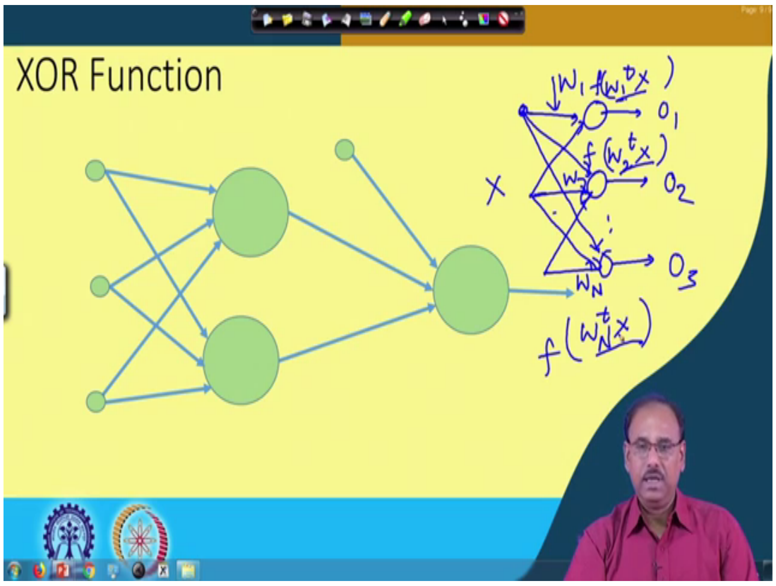


XOR Function

So, effectively the kind of neural network that I have is this. So, here I put my input to be again 1 X 1 X 2, suppose this first neuron computes OR operation. So, we know that for OR operation my weight vectors should be minus 0.5, 1 and 1 and suppose this second neuron computes NAND operation for which I have to have a weight vector as plus 1.5 minus 1 and minus 1.

So, here what I get is X 1 or X 2 and here what I get is X 1 NAND X 2; so, I use this as h 1 and this as h 2 and the second third neuron in the second layer this computes AND operation. So, here I had to have the weight vectors as minus 1.5 1 1. So, this performs an AND operation and finally, at the output what I get is XOR of X 1 X 2; so I get X 1 or X 2 at the output. So, through this discussion it is quite clear that if I have linearly separable problems, single neuron for a 2 class problem is sufficient, if I have multiple classes then I have to have multiple neurons.

(Refer Slide Time: 21:11)



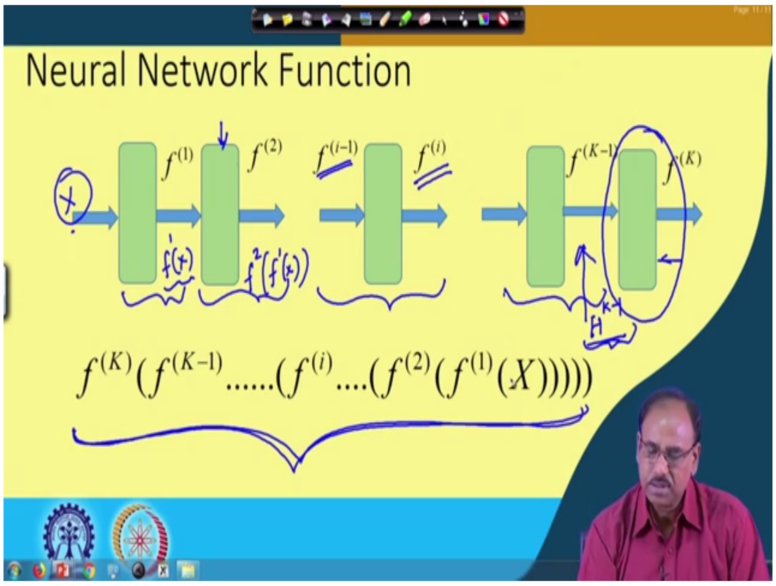
XOR Function

So, in case of multiple classes the way this has to be done is, I can have a number of neurons but in a single layer, here I have the weight vectors say something like this. So, here I have X over here we have the weight vectors, a set of weight vector for this neuron, a set of weight vectors for this neuron, a set of weight victors in this neuron.

Each of these neurons performs, suppose the weight vector for this is a W 1, weight vector for this neuron connecting to this neuron is a W 2, weight vector connecting to this neuron is a W N, if I have got n number of classes or n number of categories this neuron will compute W 1 transpose X and then a function of this; this one will compute W 2 transpose X and then a function of this, this one will compute W N transpose X and a non-linearity on this.

So, I will get a number of outputs O 1, O 2 and O 3 which actually gives you the score functions or score values for different classes or different categories corresponding to input vector X and here you find that all these are linear combinations; linear combinations of different components of the input vector. So, if my problem is a linear problem I can solve it using single layer neuron or single layer neural network, but if it is not a linear problem if it is an non-linear problem that as has been demonstrated with a very simple non-linear function of XOR function, I need multiple levels of neurons or multi layered neural network.

(Refer Slide Time: 23:25)



Neural Network Function

$$f^{(K)}(f^{(K-1)}......(f^{(i)}....(f^{(2)}(f^{(1)}(X)))))$$

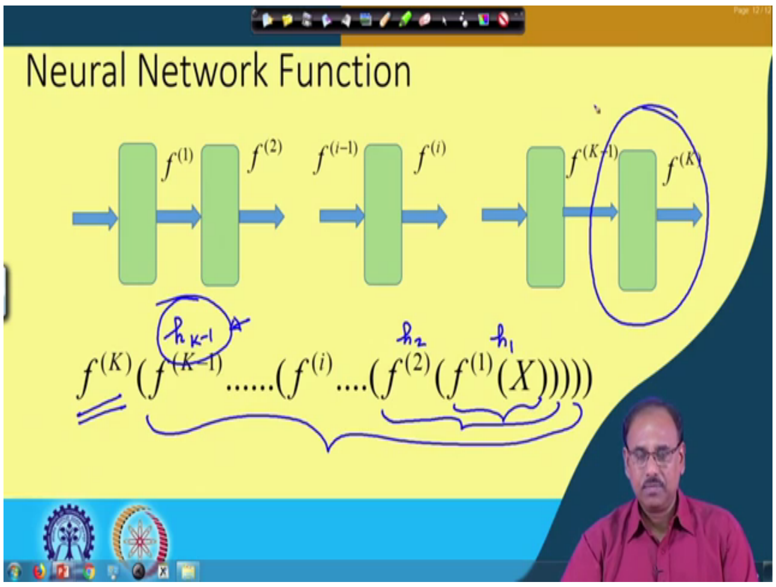
So, the kind of neural network that I had to have over here is as given by this you find that every layer of the neural network computes a non-linear function of the input feature vector of the feature vector inputted to that particular layer of neuronsand the output of that layer of neurons is an intermediate feature vector may be of same dimension or may be of different dimension depending upon how many neurons I have in that particular layer, we will see that as we proceed through our discussions.

But effectively what I have is I have a multi layer neural network and in every layer the neurons compute a non-linear function. So, when I have a cascade of this non-linear functions as given by the multiple layer neural network as has been shown in this particular diagram. So, here you find that suppose each of these are neurons a layer of neurons and in every layer it computes a non-linear functions say in ith layer it computes a non-linear function f i on the feature vectors which are computed by the previous layer given by the function f i minus 1 and each of these are non-linear function.

So, I can cascade all of them together to give you an overall cascaded function as given by this. So, if my input vector is X the first layer computes f 1 of X, this f 1 of X is inputted to the next layer. So, which computes f 2 on this f 1 of x; that means, this layer computes f 2 of f 1 of X and this way it continues when it comes over here say, I have a set of mapped featured vectors over here which is given by say h K minus 1 which is a mapped featured vector mapped from this vector X through a series of non-linear mapping, where each of these layers of neurons gives some sort of long linear mapping.

And then I have a final layer of neuron which is the K th layer which when I come to this h K minus 1 these feature vectors are non-linearly mapped from X and as these are non-linearly mapped, so this in in this feature space as given by this vector h K minus 1 these feature vectors are the problem that is posed is a linearly separable problem. And once I have such a linear problem, the final layer or the K th layer can solve this using a linear discriminator. So, effectively I can put all of them in a cascade of operation something like this.

(Refer Slide Time: 26:46)



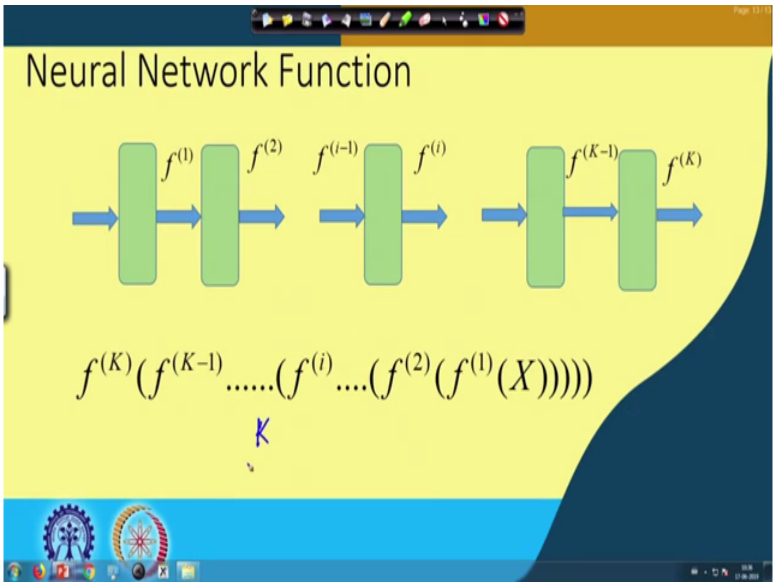
Neural Network Function

$$f^{(K)}(f^{(K-1)}......(f^{(i)}....(f^{(2)}(f^{(1)}(X)))))$$

So, at the first layer you compute f 1 of X which gives you say set of feature vectors h 1. In the second layer you compute f 2 of f 1 of X which gives you a set of feature vector say h 2. So, you find that each of them are non-linear mappings, over here through this cascading operations I get set of feature vector say h K minus 1 and when I have this h K minus 1 in this feature space as these are non-linear mapping of X, I can have a set of non-linear mapping such that h K minus 1 will lead to a linear problem or a linearly separable problemand once I have that the final 1 that is f K can be a linear machine or it might be a linear classifier. So, I have a set of operations which maps an input vector non-linearly into a space where they are linearly separable.

And once they are linearly separable then in the final step I can have a linear classifier that is what a multi layer feed forward neural network does and why is it feed forward? Because we find that the information is always flowing from input to output right? The information does not flow in the reverse way; of course, there are other variants of the neural network known as the current neural network where the information can also be fed back, we will come to that later.

(Refer Slide Time: 28:34)



Neural Network Function

$$f^{(K)}(f^{(K-1)}......(f^{(i)}....(f^{(2)}(f^{(1)}(X)))))$$

K

And here you find that the number of such cascading stages, here the number of cascading stages which is equal to K, this tells you what is the depth of the network; so, in this particular case the depth is K and as the value of K increases that is the number of layers increases that depth also increases and this is this term that is the depth of the network which actually is converted to the concept of deep learning or deep neural network ok. So, the more the depth is you are moving deeper into the problem.

So, the deep neural network or deep learning concept actually comes from the depth of the network or the depth of the network giving you the non-linear transformation it is relative to that. So, I will stop here today and I will continue with a discussion in our next class.

Thank you.