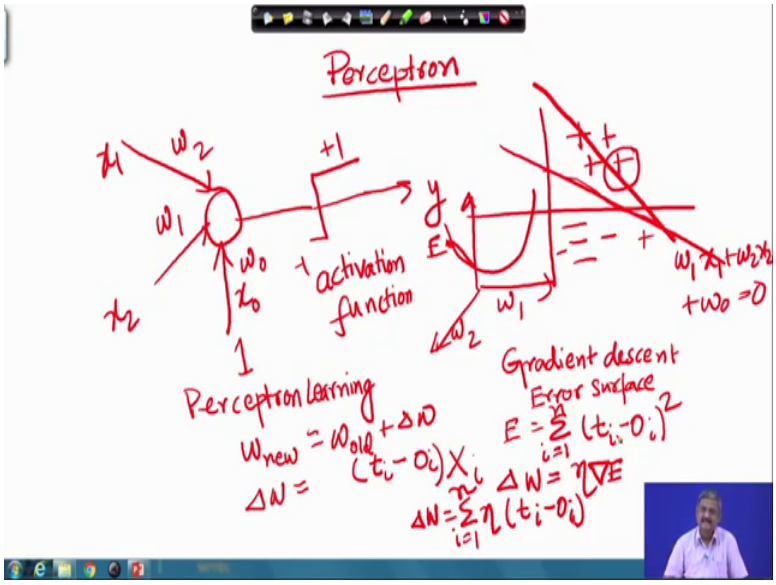
Data Mining
Prof. Pabitra Mitra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 30
Artificial Neural Networks – III

We continue on Artificial Neural Networks. To quickly recapitulate what we had done we had considered a perceptron, perceptron which takes a simple form like you have inputs let us say x 1, x 2 and you have a bias let me call it as x naught, and there are weights of this connection w 1, w naught, w 2. Then you apply a an activation function often taking the form of a step function which whose input exceeds some threshold it usually 0 it is 1 otherwise minus 1 and then that we get the final output.

(Refer Slide Time: 00:26)



We also discussed that this is equivalent to drawing the perceptron actually realizes a linear line 2 plus by 2 points and w naught, w 1 are nothing, but the biases and the slopes of this line. So, it actually implements a line of this form implements a line of this form. And then we studied, so what we do to given in prediction problem where you are given a training set consisting of n number of input vectors and their corresponding desired outbreak output vectors.

What we do is to start with random values of the widths w naught, w 1, w 2 and then update the weights over iterations till the weights become stable they do not change. And

we discussed two update rule 1 is the perceptron learning rule where what we do is to start with a random weight and then look at a training example. If the every set of weights random or whatever will correspond to a line if that line correctly classifies that training point do not do anything maxim, if it misclassifies you tilt the line you tilt the line to try to classify it.

So, what is the tilting rule? Your w new is w old plus some delta w where delta w equals the target output for that x i let us say call it t i and the actual output for that x i let me call it o i. So, given the value of weights for the input I collected, I calculate actual output y and I am supposed to achieve some target t i just multiply that by your x i multiply that by x i and this correction you go on making to w till all points are correctly classified.

We can actually show that if the points are from 2 classes are linearly separable this algorithm will indeed converge. Then we also studied an alternate algorithm called the gradient descent algorithm what the gradient descent algorithm does for differing different values of w naught, w 1, w 2 it calculates an error surface it calculates an error surface.

So, what is the error surface? The error surface is nothing but if you sum up over all the training points it is the target output minus the obtained output square, mean squared error summed up over all the training examples in number of training examples. So, and then we can visualize as if we look at the w 1 w 2 plane for example, and plot the error in the y axis we will get some surface like this.

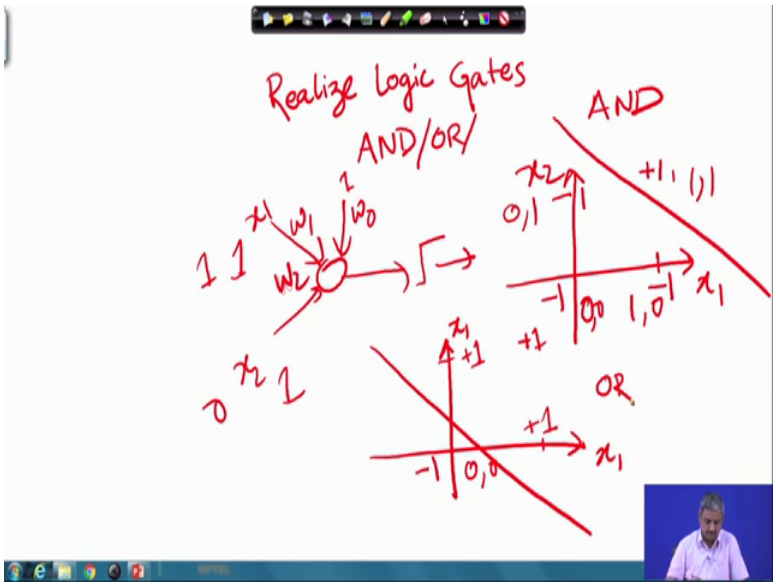
And what the gradient descent does is that it starts with a random value of w 1, w 2, w naught or even more w's are there then does what is called a steepest descent; that means, it changes the value of w 1, w 2, w naught which leads to maximum decrease in the error. They are defined like this it goes takes a direction along, which is same as descending along the tangent of the error surface.

So, delta w is some learning constant eta times the tangent of the derivative of the error function which I represent by this symbol. For this kind of mean square error thing error you can actually show that delta w becomes nothing, but eta times t i, o i summed over all the training examples because the error is itself is some about all returning examples.

So, in the inverse slides that derivation are there I am not repeating the derivation. So, if you do it you get a real tie blue. So, that they would weight update is same in perceptron as well as the gradient descent algorithm only difference being perceptron it looks at every example if it is misclassified it updates and works with a updated value of w whereas, the gradient descent calculates delta w based on the sum of the correction over all the training points then update stop.

So, 1 is a incremental mode every training example changes w other is a batch mode you go through all the examples then change w. So, both of these does converge for the simple case of a perceptron.

(Refer Slide Time: 07:19)



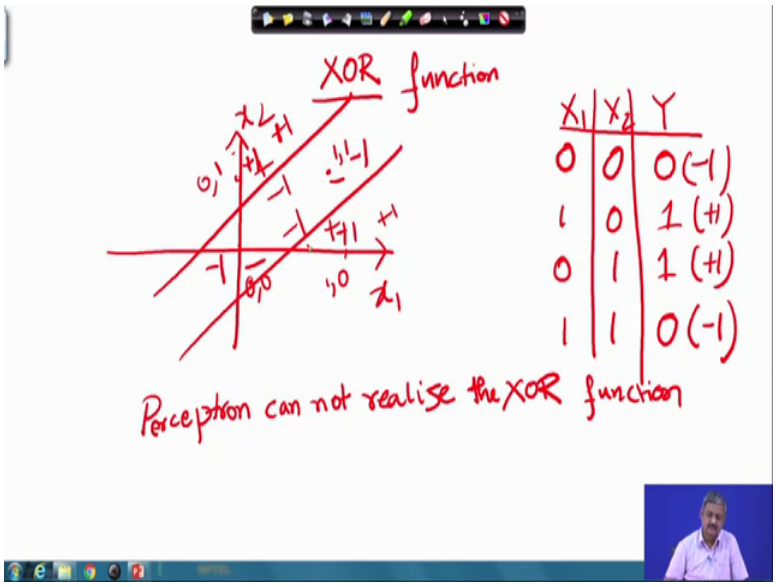
And we also showed that you can realize logic function. So, logic gates like and or etcetera by choosing proper value of that weights. So, putting proper value of weights and the bias put with the threshold function would realize these gates. So, I would request you that you take some standard AND, OR functions and you try to find out what are the values of w 1, w 2 which will give exactly the output that this logic function should give for inputs like 1 1, 1 0 and so on.

What is the value of w that would give the proper input for the given logic gate. That is in fact, easy to understand in some sense because if we say take 2 input logic functions x 1, x 2 and for example, if I take the AND function. So, if I draw which at the points for which and is true you will see that 1 1 is and is true, rest all point 1 0 0 0 0 1 are negative

examples and is false. So, this is the AND function in x 1 x 2 space. So, this is 0, this is 1 this is 1 1, this is 0 1, this is 1 0, this is 0 0. So, this is the AND function. And since the perceptron is a straight line you can actually draw a straight line to realize the 1 1 in one side and the rest of the point in other side.

So, similarly you can see the OR function. So, what is the OR function? So, you have 0 0 as the negative example and rest as the positive example true examples where OR is true. Similarly, you can draw a line to recognize the OR, so this is OR function; so many of the gates. But now let me give you an example of a logic gate which cannot be realized by this perceptron.

(Refer Slide Time: 10:11)



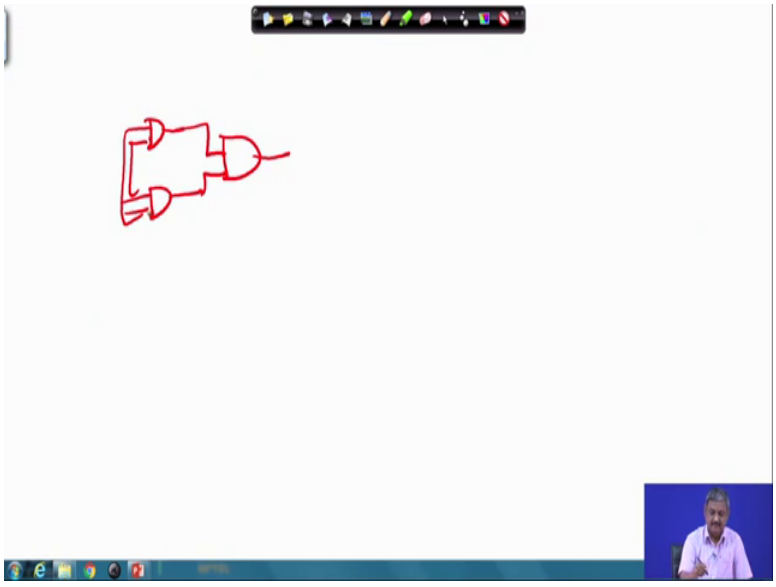
Let us look at the XOR gate. Let me write down the truth table. Let me call 0 as 1 classification problem, 1 1 XOR is 0. So, if I plot it, you can see these two are the negative points 0 0 and 1 1, and these two are the positive points 1 0 and 0 1. So, this is negative whereas, this is positive.

Now, you cannot draw a line to separate the positives from the negatives, no line, you can check no line can be drawn. So, the perceptron cannot XOR function cannot realize the XOR function. So, what is the solution? Perceptron failed because I do not I cannot separate them by a line, but maybe I can separate them by 2 lines. So, maybe I can separate them by say 1 line like this and another line like this. So, everything inside this 2 line is minus outside it is plus, maybe I can draw. So, or is inside of both the line is

plus and non origin side sorry non origin side of both the line is a plus point for XOR and origin side of both the line is minus point for x. So, if I can use 2 lines I can realize it all right.

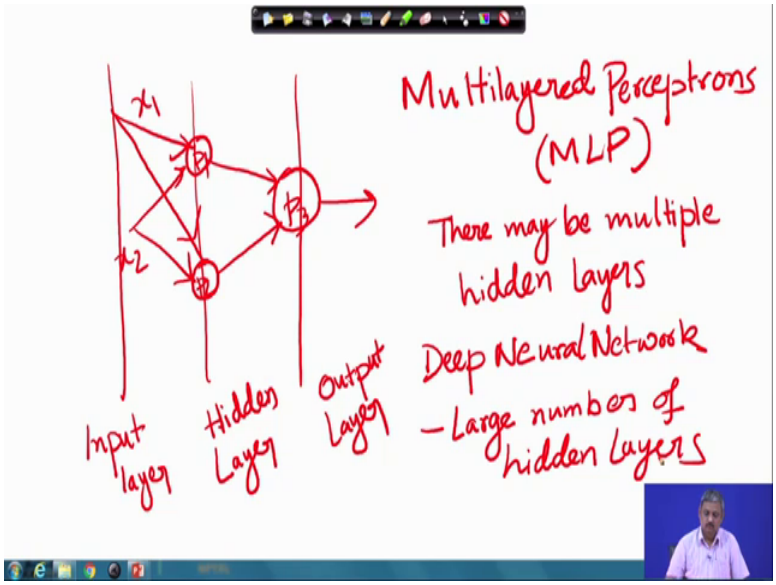
So, then people thought, so how do you realize two or more lines. Let me connect up perceptrons. So, it is like you remember in your logic circuit maybe a single AND function cannot realize. So, what you do is that you connect up and gates I hope you are familiar with digital logic and connect a second level up gate.

(Refer Slide Time: 13:29)



Now, you can realize this, now you can realize this all right.

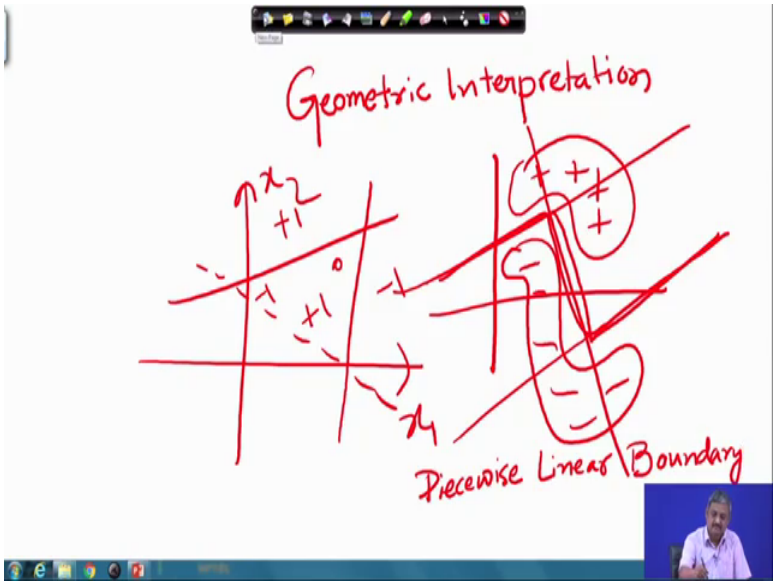
(Refer Slide Time: 13:44)



So, the same idea if not a single perceptron if not a single perceptron you add up 2 perceptrons this is perceptron 1 and perceptron 2, and the output of these two perceptrons you feed as input of a third perceptron. So, earlier inputs output of the single perceptron as the output, now I have 2 perceptron and their output goes to the third perceptron as input.

And output of the third perceptron is our actual output. So, this type of networks are called because there are different layers of perceptron you can think of one layer feeding to the next layer, feeding to the next layer, these are called multi layered perceptron or MLP, they are called multi layer perceptrons or MLP. So, and this, the terminology is like this the first inputs are called input layers input layer the second level perceptrons are called hidden layer and the output is called the output layer. So, there may be multiple hidden layers as many as you want. In fact, the modern trend is something called a deep neural network which is nothing but you have large number of hidden layers that is why it a (Refer Time: 16:27), large number of hidden layers.

(Refer Slide Time: 16:46)

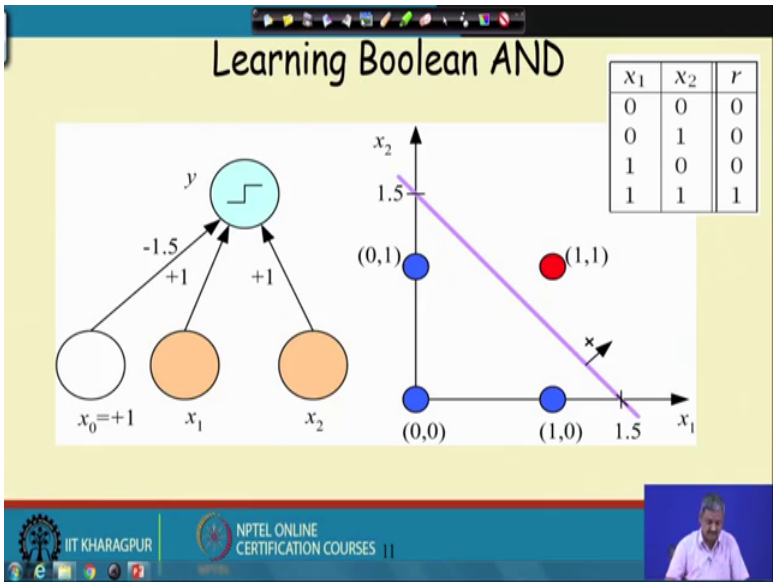


So, what does this mean geometrically? So, each layer its inputs or each perceptron would give you a line like this some w 1, w 2 on another perceptron would give and the output of these perceptrons would be by either plus 1 or minus 1 depending on which side of the perceptron the point to be classified first. And this value plus 1 minus 1 they are input to a third second level perceptron which look at if it is in plus side of both the lines or plus side 1 line minus side of other line that the third perceptron looks at and gives the result.

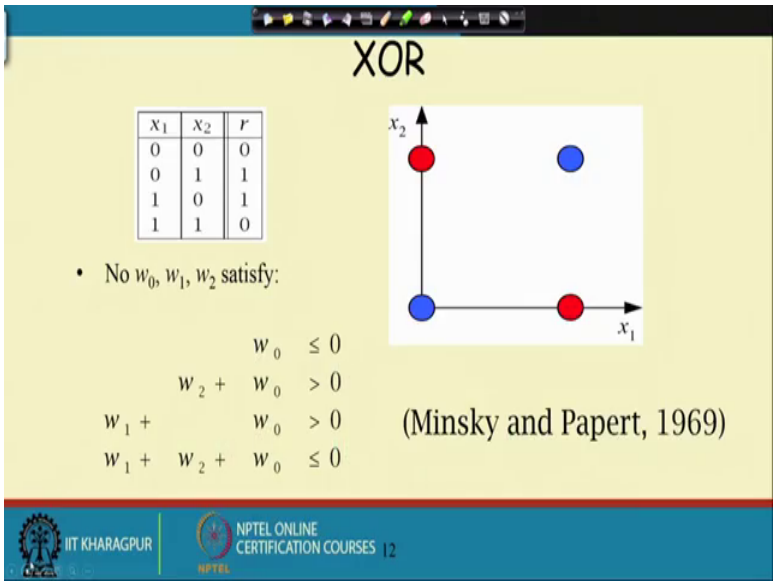
So, then you can see many non-linear non-linearly separable set of classes for example, if the classes are like this you can sort of classify them by considering a number of pereptrons. So, these is a boundary piecewise linear boundary you would call it. In fact, it can be shown that any complex surface can be realized by such things.

So, I have explained you everything with the help of a logic functions, but in general it holds true for any kind of classification task.

(Refer Slide Time: 18:39)

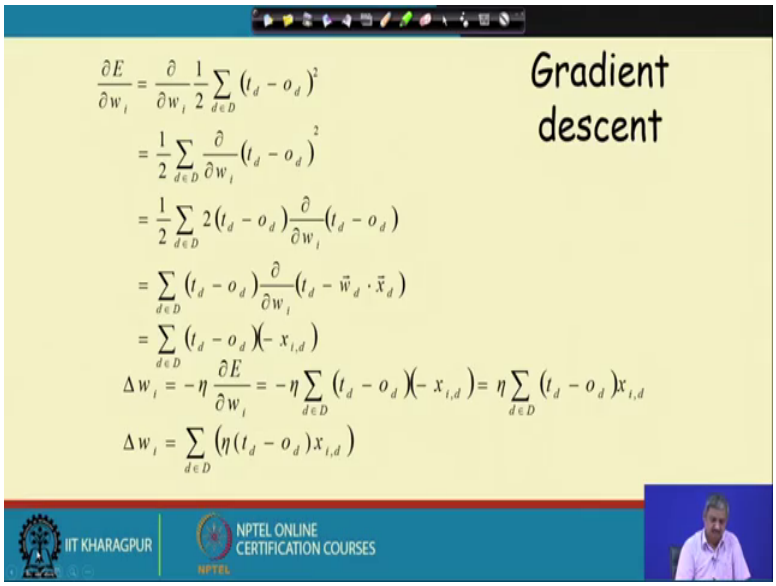


(Refer Slide Time: 18:48)



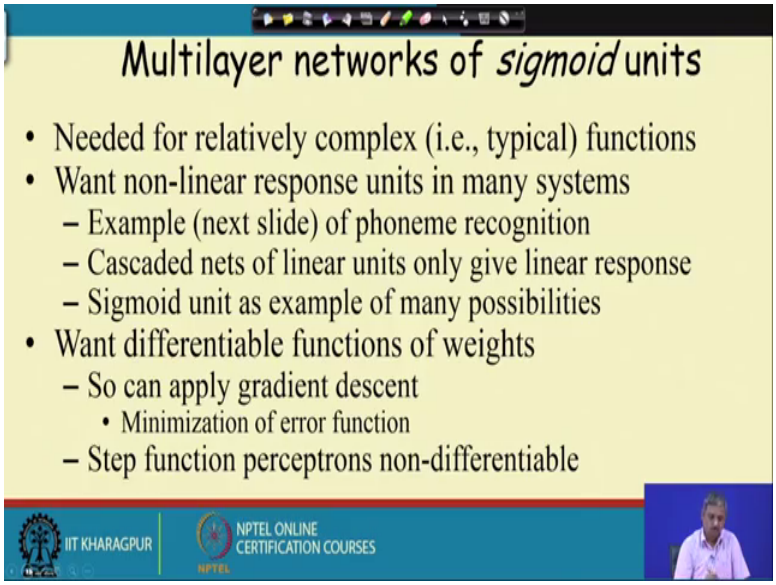
So, this is the XOR, this is the XOR, I will skip this perceptron learning rules I have already told you all this gradient descent error minimization.

(Refer Slide Time: 19:01)



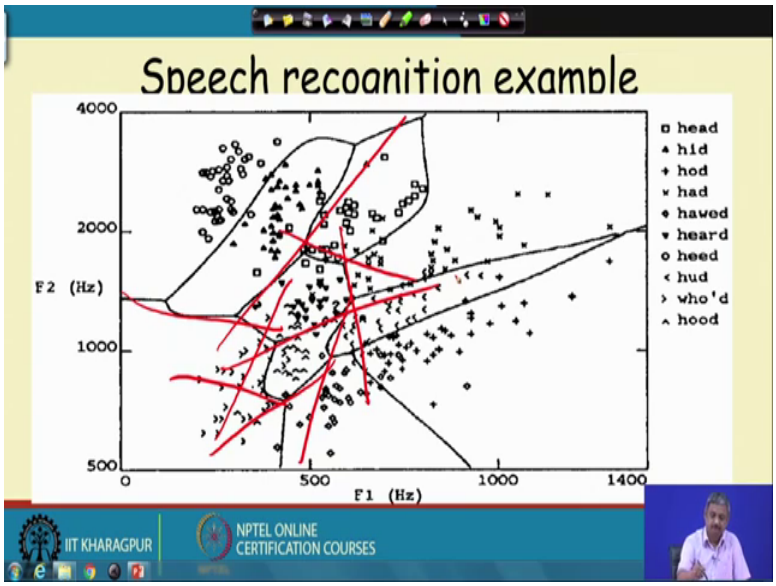
You should go through this slide for the derivation that I talked about of the update rule, of the update rule.

(Refer Slide Time: 19:16)



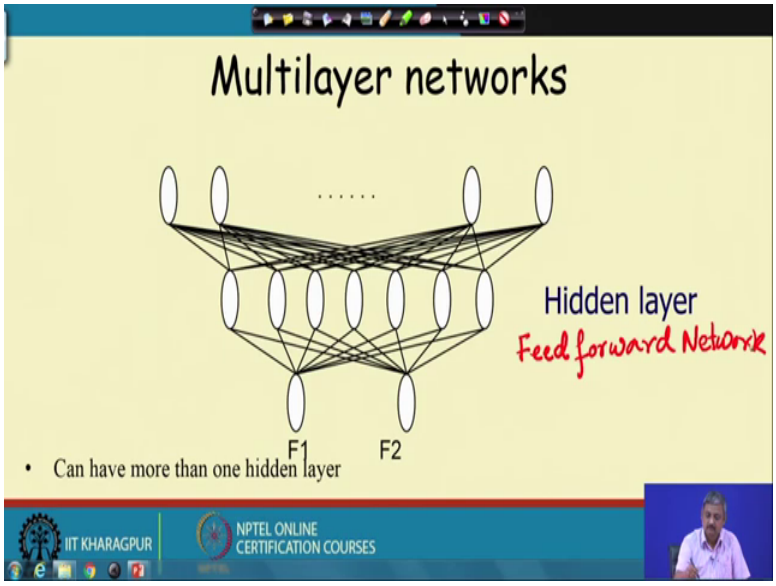
So, if you have a complex function for example.

(Refer Slide Time: 19:20)



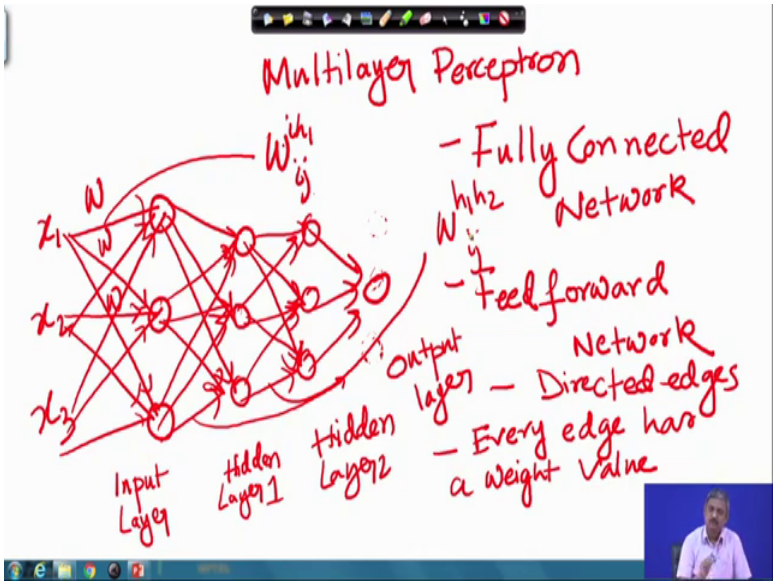
this is an problem, this is a speech recognition problem. So, this for you have to recognize the vowels the 5 vowels, the 5 vowels and the if we look at the frequencies of utterances of the speech signal for these vowels they would form a complex class boundary like this, no sorry, not vowel it is some a set of words head hid and a hod, had, howed similar words ok. Even actually the vowel also looks like this. So, you can draw a multi layer perceptron to sort of separate them out.

(Refer Slide Time: 20:20)



As I have told this is the, please note down the structure this is the multi layer structure. This is also called a feed forward network.

(Refer Slide Time: 20:57)



Actually let me clearly draw the, make the picture clear. Suppose I have 3 inputs each of the input will be collected connected to all the neurons, this diagram is important. Each input corrected to each of the input neurons, this is the hidden layer say 1. The output of each of the input neurons let me put arrows here will be connected to all the nodes in the next layer. So, this connects to all the nodes in the next layer, this connects to all the nodes in the next layer this connects to all the nodes in the next layer.

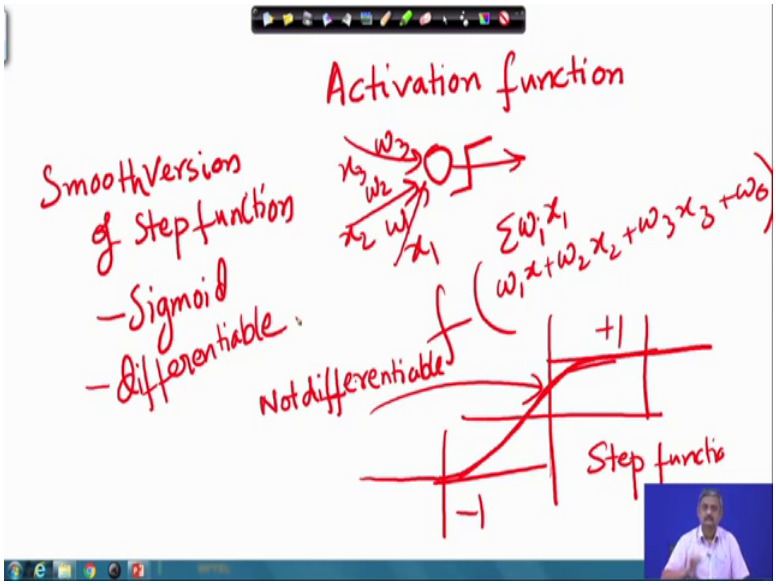
Similarly, you can have a hidden layer 2, same rule holds it is connected to every node in the next layer and finally, the output. You can have less number output for example, you can have only one output which gives plus 1 or minus 1. You can have only one output which would give plus 1 or minus 1. So, this is the output neuron. Again same rule will be everything will be connected.

So, two things you observed, first thing it is a fully connected network, every node in a layer connects to all the nodes in the next layer. Second thing every node connects to only nodes up next layer providing inputs to only node of next layer, there is no backward connection. So, nodes of this layer contact as input of the nodes of a previous layer. So, this would be called a feed forward network this is a feed forward network. Because it feeds forwards no back link. There is another class of network which has back

links which are called recurrent networks which is also an very important class of networks, but I am not covering that right now.

So, this is a network and then you have weights, each of these links would have weights. So, so you have directed links 1 output of 1 goes as input of another you have directed edges and every edge has a weight value every edge has a weight value. The convention of writing the weight is something like this, if you write W ij say input to h 1 it means it connects the ith neuron of the input layer ith perceptron of the input layer to the jth perceptron of the hidden layer 1, that would be this. Similarly this would be, similarly this would be w h 1, h 2, ij; that means, ith neuron of the hidden layer 1 connects to j th neuron of hidden layer 2.

(Refer Slide Time: 26:16)



So, now you can one more thing is missing that, I have talked about weights, but each of this neuron just like a perceptron we will have activation function. So, each of this neuron just like a perceptron will take inputs, weighted inputs and add them up, so it will compute w 1, x 1, v plus w naught the bias term and add I apply a activation function of this which usually takes a form either plus 1 or minus 1. To step like a function step function that is the activation function that.

But one problem with this step back function it is not differentiable here, not differentiable at the corner of the step. But when you are taking gradient descent tangent means some kind of derivative has to be taken differentiation has to be done. So, people

give a smooth version of step function is the sigmoid function which approximates this step by a s, like function s, that is in between if outside it is same as the step, but here there is a is of a jump there is a smooth transition and differentiable everywhere. It is now differentiable which will need for the later stage all right. So, this is clear, the multi layer perceptron.

In the next lecture we will discuss just like the perceptron weight update rule and weight update rule for the multi layer perceptron.

Thank you.