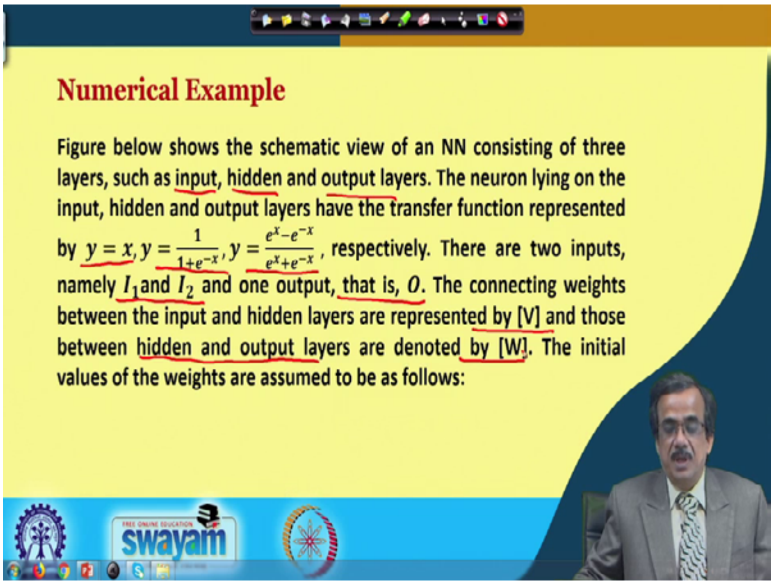
Fuzzy Logic and Neural Networks
Prof. Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

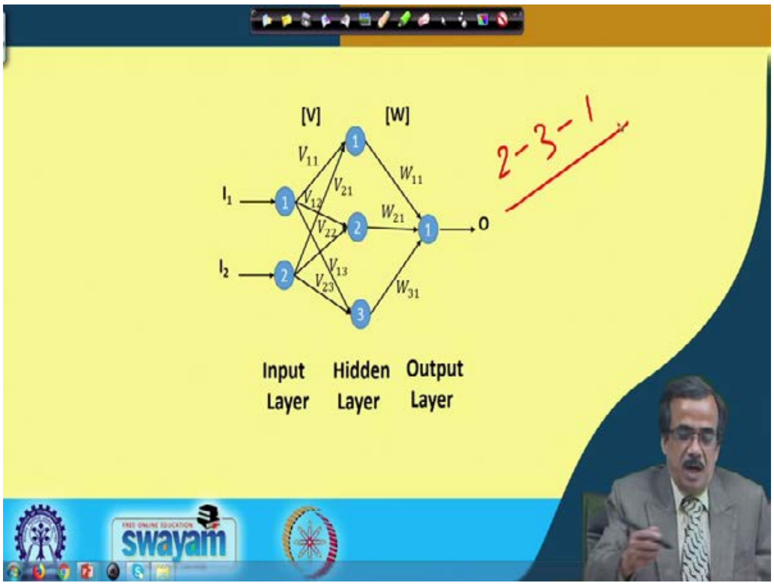
Lecture – 24
Some Examples of Neural Networks (Contd.)

(Refer Slide Time: 00:14)



Now, we are going to discuss, how to solve one numerical example related to the multi layered the feed forward network.

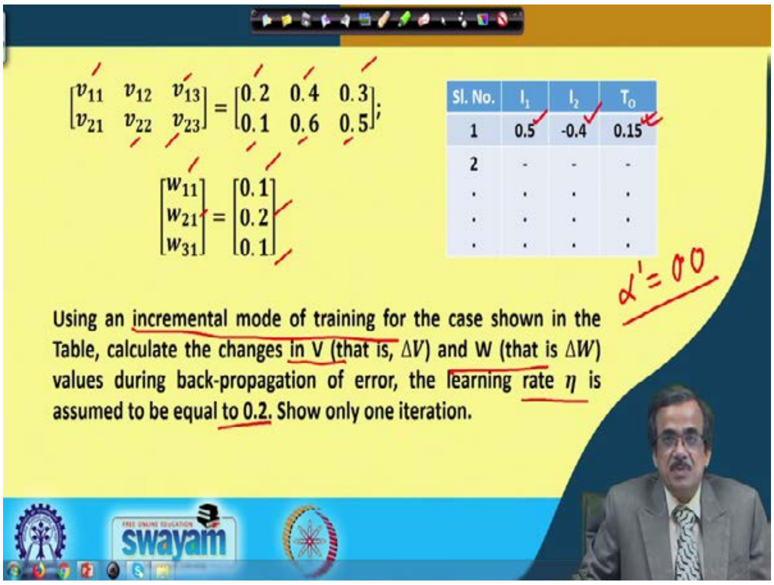
(Refer Slide Time: 00:36)



Now, here I am just going to show a small network. So, before I read this particular statement, I am just going to show a small network. Now, this network is nothing, but actually a three layered network, now on the input layer, we have got say 2 neurons, on the hidden layer, we have got the 3 neurons, and on the output layer actually, we have got 1 neuron.

So, this is nothing, but a 2-3-1 network. And, now, I am just going to show and I am just going to state the problem. So, this is the schematic view of this multi layered feed forward network and it consists of three layers like your input layer, hidden layer and output layer. The neurons lying on the input, hidden and output layers have the transfer functions represented by $y = x$ on the input layer (that is a linear transfer function), $y = \frac{1}{1+e^{-x}}$, (this is nothing, but is your log sigmoid transfer function) and $y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (this is nothing, but the tan sigmoid transfer function), respectively. There are two inputs: I_1 and I_2 and there is only one output, that is, O. The connecting weights between the input and the hidden layers are denoted by V and that between the hidden and output layers are denoted by the W.

(Refer Slide Time: 02:37)



Now, the initial values for these particular connecting weights are shown here. Now, here you can see that is v_11 is nothing, but your 0.2; that means, your the connecting weights between the first input neuron and the first hidden neuron, that is, v_11 is 0.2.

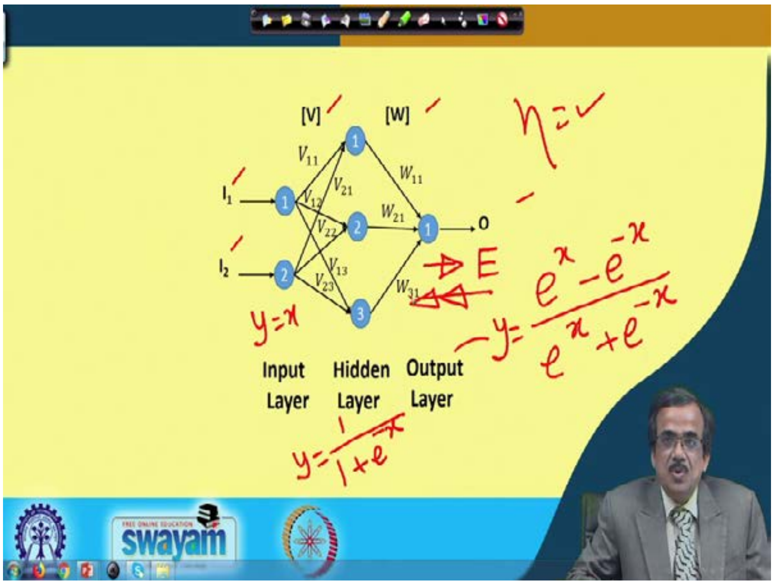
Similarly, v_12 is 0.4, v_13 0.3, v_21 (that is between the second neuron lying on input layer and the first neuron lying on the your the hidden layer) is nothing, but is your 0.1, v_22 is 0.6 and v_23 is nothing, but 0.5.

Now, similarly, the connecting weights between the hidden layer and output layer, that is w_11 (that is the connecting weight between the first hidden neuron and the output neuron) is 0.1. The connecting weight between the second hidden neuron and output neuron, that is w_21 is 0.2, similarly, w_31 is equal to 0.1 and here, you have got a large number of training scenarios and out of all the training scenarios, supposing that say I am just going to show only one.

Now, the training scenario is something like this, if I_1 is 0.5 and I_2 is minus 0.4, then the target output is nothing, but 0.15, now we are going to use the incremental mode of training and using this incremental mode of training, we are going to find out, what should be the modified value for this V and the modified value for this particular your the W.

So, our aim is to determine the changes in the values of V and W during this training and we are going to consider the learning rate, that is, η is 0.2 and for simplicity, actually the momentum constant that is α' has been taken to be equal to 0.0; that means, we did not consider the momentum term. And, through hand calculations, we are going to show one iteration of this particular network. Let us see how does it work. Now, before I go for so, this particular solution let me once again look into this particular network. So, it is a very simple network.

(Refer Slide Time: 05:37)

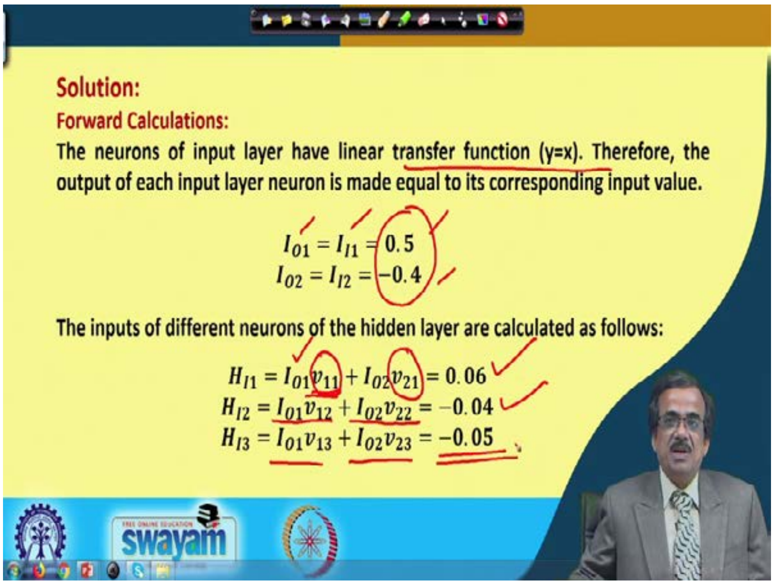


So, what you do is, here we have got 2 inputs and 1 output, these are the connecting weights, and here, we have got the transfer function like $y = x$ and in the hidden layer actually, I have got the log sigmoid transfer function, that is, $y = \frac{1}{1+e^{-x}}$. In output layer, we have got the tan sigmoid transfer function that is nothing, but $y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

And, this learning rate value we have assumed and the moment I pass say one set of training scenario, I will be able to find out, what is the calculated output. Now, this calculated output will be compared with the target and the error will be determined and this error will be propagated back for the purpose of updating the connecting weights, so that this particular network can predict the say the output for a set of inputs more accurately.

Now, let us see, how to carry out so this particular calculations and how to find out the change in V and your the change in W values in order to minimize the error in prediction.

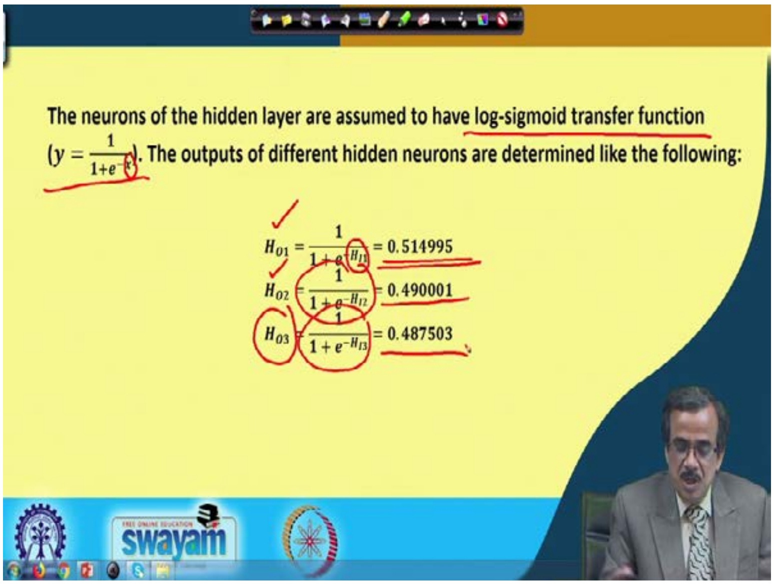
(Refer Slide Time: 07:02)



Now, the way it has to be solved, I have already discussed, now let me repeat. So, what we will you have to do is, in the input layer, we are using the linear transfer function of the form $y = x$. So, output will be nothing but the input. Now, here the same symbol, I am just going to use the same nomenclature, for example, say I_O1 is nothing, but the output of the first neuron laying on the input layer, I_I1, that is, your input of the first neuron lying on the input layer is nothing, but 0.5, similarly, I_O2 is nothing, but I_I2 is nothing, but is your minus 0.4.

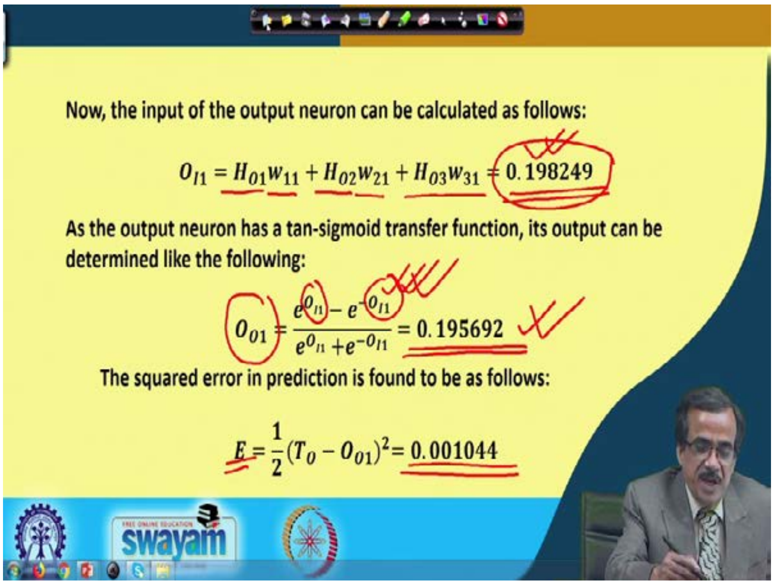
So, these are nothing, but the outputs of this particular input layer, and once you have got this particular output, now the respective outputs actually we are going to multiply by the connecting weights and we can find out like what should be the input of the different neurons lying in the hidden layer. For example, say H_I1 that is input of the first neuron lying on the hidden layer is nothing, but I_O1 multiplied by your v_11 plus I_O2 multiplied by v_21 and if you calculate, you will be getting 0.06. Now, similarly, this H_I2 is nothing, but I_O1 multiplied by v_12 plus I_O2 multiplied by v_22 and that is nothing, but minus 0.04, and similarly, I can find out H_I3 that is nothing, but the input of the 3rd neuron lying in the hidden layer and that is nothing, but I_O1 multiplied by v_13 plus I_O2 multiplied by v_23 and that is nothing, but minus 0.05. And, once you got these particular inputs of the hidden neuron, now very easily, we can find out what should be the corresponding output.

(Refer Slide Time: 09:54)



Here, we have got this particular transfer function, that is your the log sigmoid transfer function and that is nothing, but $y = \frac{1}{1+e^{-x}}$. So, this particular x is actually I will have to put the input of the different hidden neurons. Now, this $H_{O1} = \frac{1}{1+e^{-H_{I1}}}$ and if you put this numerical value and solve there is a possibility that I will be getting this particular the output.

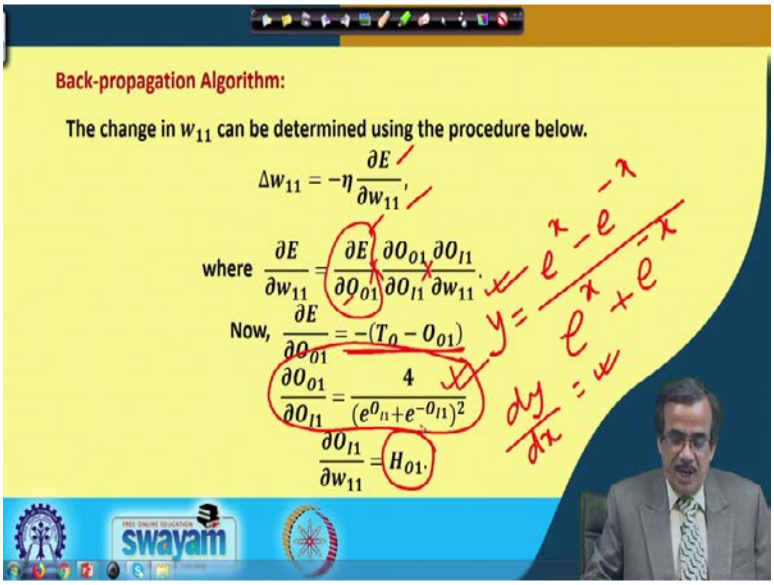
Similarly, the output of the second neuron lying on the hidden layer using this particular expression I can find out that is your 0.490001 and by following the same, I can also find out what is H_O3, that is output of the 3rd neuron lying on this particular hidden layer and this is nothing, but your 0.487503. So, this is the way, actually we can find out what should be the outputs of your different hidden neurons.



And, once you have got, this particular output, we can find out, what should be the input of the neuron lying on the output layer. So, this O_I1 is the input of the first neuron lying on the output layer, we have got only 1 neuron lying on this particular output layer. So, $O_{I1} = H_{O1}w_{11} + H_{O2}w_{21} + H_{O3}w_{31}$. And, if you insert the numerical values and calculate, you will be getting this is nothing, but the calculated output of this particular network for this set of inputs, and once you have got. So, I am sorry. So, this is nothing, but the input of the neuron lying on the output layer.

So, if I know this particular input, I can find out what should be the output of this particular neuron lying on the output layer and here, actually we have got the tan sigmoid transfer function. And, for this tan sigmoid transfer function, this O_I1 is nothing, but this the input of the neuron lying on output layer. So, I will be getting the calculated output of the neuron lying on the output layer is nothing but this O_O1. Now, if you know this calculated output so, very easily we can find out what is this error. So, this error in prediction is nothing, $E = \frac{1}{2}(T_O - O_{O1})^2$ and if you calculate, we will be getting this as the error and based on this particular error, actually I will have to do the updating.

(Refer Slide Time: 13:27)

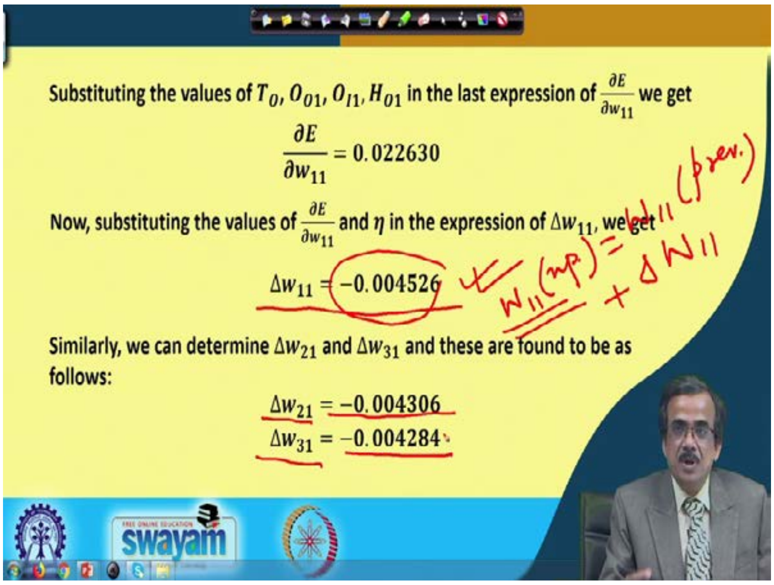


Now, let us see how to update. So, this particular connecting weight, that is your w_11, now this w_11 is actually nothing, but is your this particular thing. So, this is your w_11. So, I am just going to update it. So, I am just propagating back this particular error and I am going to update this particular connecting weight (Refer Time: 13:55). Now, let us see how to update this particular connecting weights. Now, to update the connecting weights, we are using, in fact, the back propagation algorithm or the delta rule now according to this delta rule, the change in w_11; so, $\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}}$.

Now, the partial derivative of E with respect to w_11 is nothing, but partial derivative of E with respect to O_O1 multiplied by the partial derivative of O_O1 with respect to O_I1 multiplied by the partial derivative of O_I1 with respect to your w_11. Now, here, we have already discussed like how to find out this partial derivatives, for example, say your this partial derivative of E with respect to O_O1, very easily you can find out this particular expression, then comes your this partial derivative of O_O1 with respect to your O_I1 and here, we have got actually the tan sigmoid transfer function.

Now, if you write down the expression for tan sigmoid. So, $y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. So, very easily actually, we can find out what is dy/dx, this I have already discussed. Now, if you find out the dy/dx, then with a little bit of simplification so, you will be getting this particular expression. So, this is nothing, but the partial derivative of O_O1 with respect to O_I1 is nothing, but this then partial derivative of O_I1 with respect to w_11 is nothing, but H_O1 and once you have got all the terms.

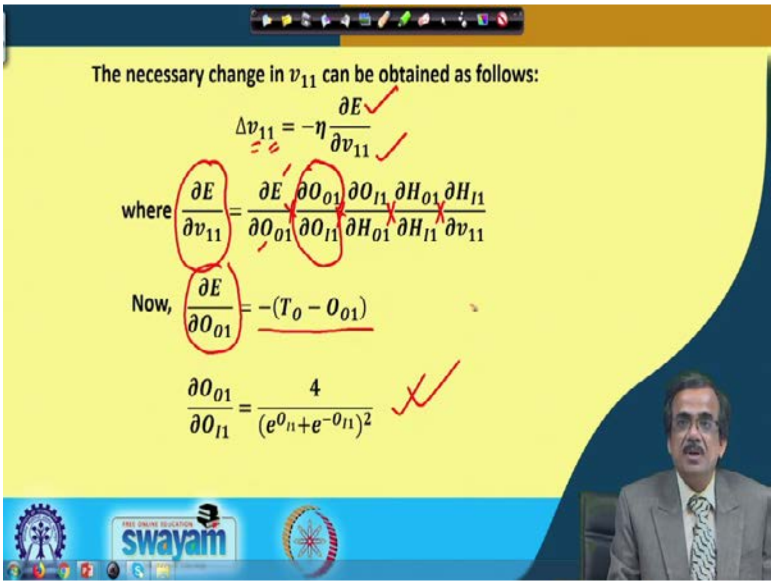
(Refer Slide Time: 16:22)



So, now actually, what you can do is, we can multiply just to find out, what is this your partial derivative of E with respect to your w_11 and we also put actually the numerical value of the learning rate. And, once you have got this particular thing, very easily we can find out, what should be this change in your w_11, that is change in w_11 is nothing, but minus 0.004526 and once you have got it very easily you can find out your w_11 updated is nothing, but w_11 previous plus your Δw_{11}.

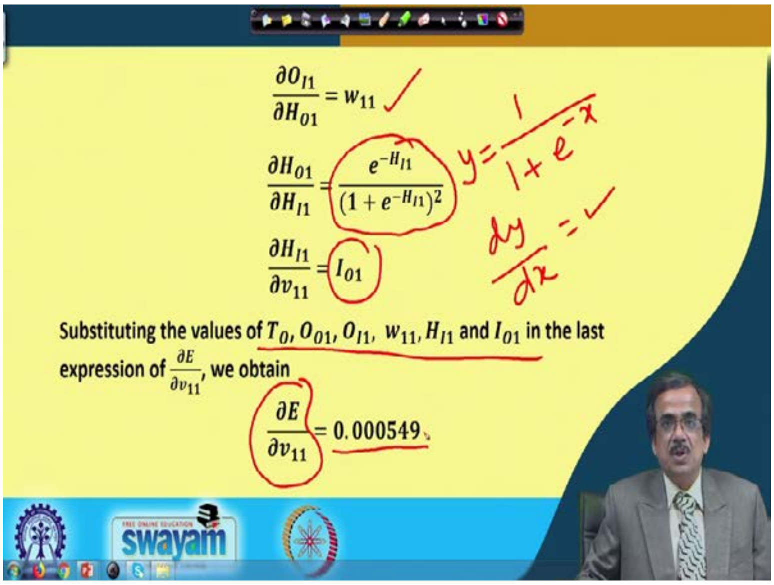
Now, this Δw_{11} we have already got. So, very easily, you can find out the updated value for this w_11. Now, the same principle we are going to use for determining, what should be the updated value or what should be the change in w_21. So, change in w_21 will be minus 0.004306, then change in w_31 is nothing, but minus 0.004284. So, by using the same principle, we can find out what should be the change in w values.

(Refer Slide Time: 17:53)



And, once you have got, now, you will have to find out the change in v values. So, this v_11 is nothing, but the connecting weights between the first neuron of the input layer and the first neuron of this particular the hidden layer. So, $\Delta v_{11} = -\eta \frac{\partial E}{\partial v_{11}}$. Now, the partial derivative of E with respect to this v_11 is nothing, but your partial derivative of E with respect to O_I1 multiplied by the partial derivative of O_O1 with respect to O_I1. Then comes your the partial derivative of O_I1 with respect to your H_O1. Then, partial derivative of H_O1 with respect to H_I1, then partial derivative of H_I1 with respect your v_11 and once again we are going to use the chain rule of differentiation. Now, these partial derivative of E with respect your O_O1. So, I can find out this particular expression, then comes your the partial derivative of O with respect to your O_I1 this we have already seen. So, we can find out.

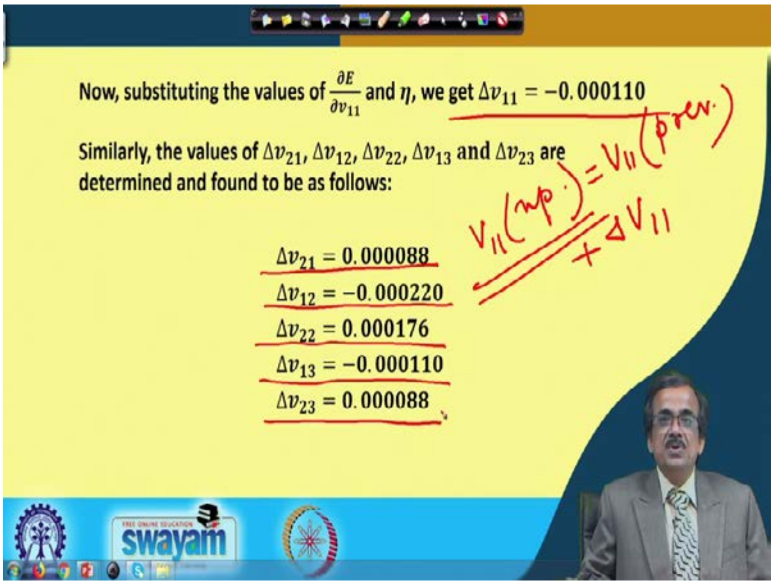
(Refer Slide Time: 19:33)



The next is your the partial derivative of O_I1 with respect to H_O1 is nothing, but w_11, then partial derivative of H_O1 with respect to H_I1 is nothing, but e raise to the power minus H_I1 divided by 1 plus e raise to the power minus H_I1 square. Now, this is nothing, but is your the log sigmoid transfer function and it is of the form $y = \frac{1}{1+e^{-x}}$.

And, we have already discussed that this particular derivative can be determined and once you have got this particular derivative with a little bit of simplification, you will be getting this particular expression and your the partial derivative of H_I1 with respect to v_11 is nothing, but I_O1 and once you have got all the expressions, now we can put together and we can write down. So, this expression of partial derivative of E with respect to v_11 and if you put all such numerical values in that particular expression; you will be getting the partial derivative of E with respect to v_11 is nothing, but 0.000549.

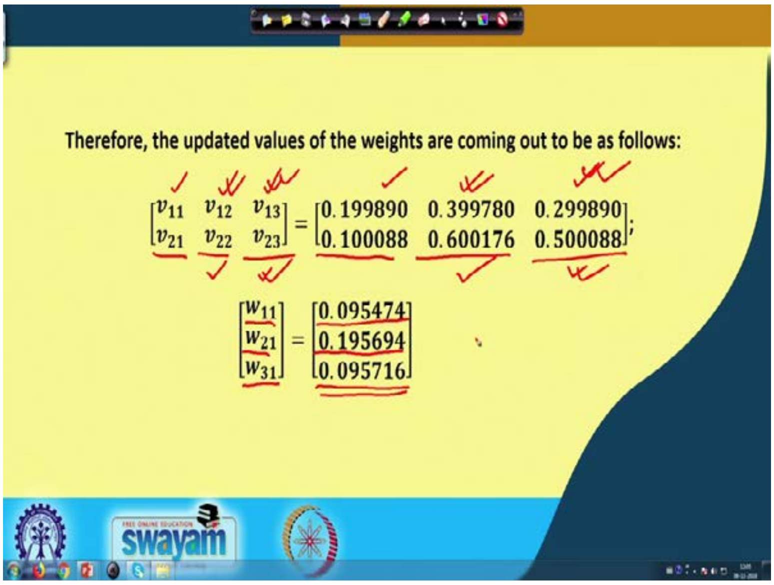
(Refer Slide Time: 21:12)



And, once you have got this very easily you can find out what is your Δv_{11}. So, this Δv_{11} actually, we can find and once you have got this Δv_{11}, now we can find out what should be the updated value for this is your v_11 because v_11 updated once again is your nothing, but v_11 previous plus your Δv_{11}.

And, we can find out, what is this v_11 updated and once you have got by following the same principle, we can find out like what should be your the change in v_21. Now, the change in v_21 is something like this then change in v_12, change in v_22, change in v_13 change in v_23. So, all such numerical values, we can find out.

(Refer Slide Time: 22:22)



Now, actually, we are in a position to find out the updated values for this particular network in one iteration , now if you see the updated values. So, the updated value for this v_11 will become 0.199890. Similarly, the updated value for v_12 is nothing, but this v_13 is nothing, but this, then comes your v_21 updated value is this, v_22 the updated value is this and v_23 the updated value is something like this. And, I can also find out the updated values for this w that is your w_11, the updated value will be something like this, for w_21 the updated value will be something like this, and for w_31 the updated value for this something like this.

Now, once you have got the particular updated values and I am using say the incremental mode of training. So, I can find out this updated values and using the updated values once again, if I pass the same set of training scenario, there is a possibility that I will be getting a slightly less error in prediction and supposing that I am running for say 10 or 20 iterations by following the same principle before I go back or before I start with the second training scenario.

So, based on the first training scenario, let me update for 10 times or let me just run this for say 10 times, 10 iterations, then we go for the second training scenario and repeat the process. Then, you go for the third training scenario, you repeat the process and all the training scenarios you pass one after another and at the end of each training passing each training scenario, you update this particular network.

Now, if you follow this particular method, there is a possibility that you will be getting one network, the optimal network or the near optimal network, after passing the 10-th training scenario and whatever you got after passing the first training scenario, there could be a lot of difference. So, these two networks could be different performance-wise and if you follow this incremental mode of training, there is a possibility that you may not get a very good generalization capability of this particular network.

The network may not be adaptive in nature and if it is not adaptive in nature, for the unknown test scenario, this particular network may not work well. Particularly, if you just go for the incremental mode of training, which is computationally very fast compared to the batch mode of training, but its generalization capability may not be sufficient.

Thank you.