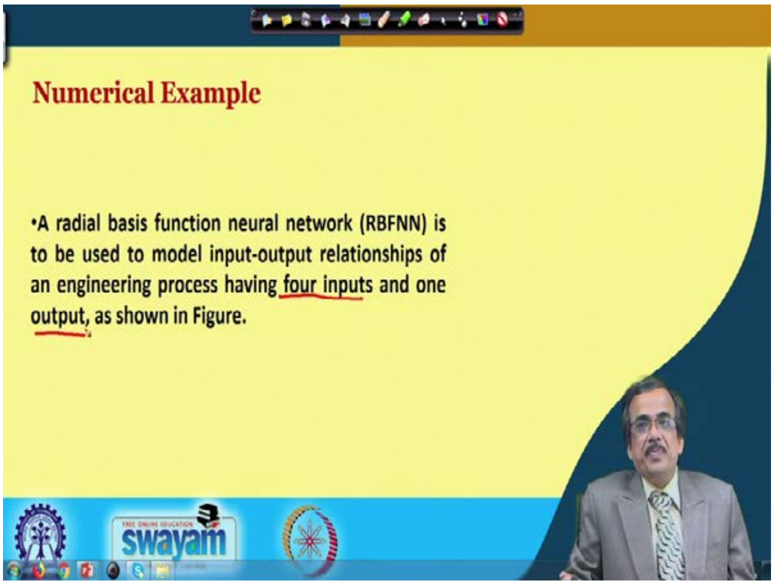
Fuzzy Logic and Neural Networks
Prof. Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

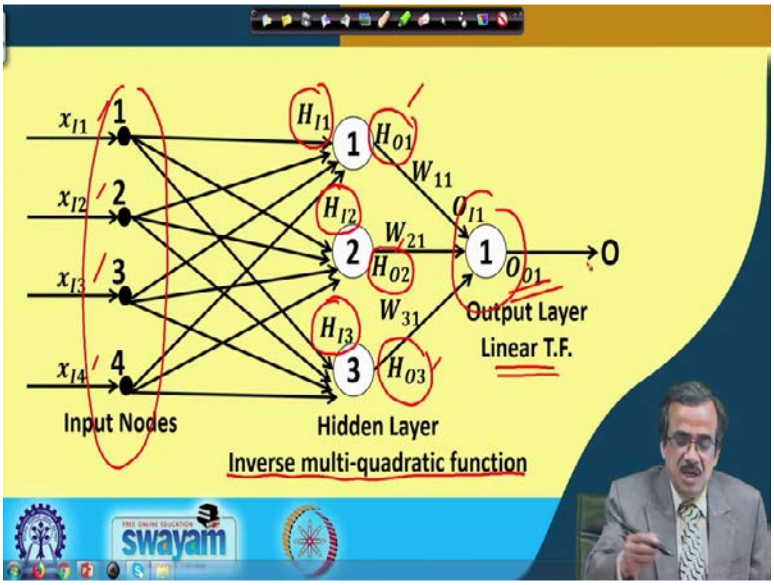
Lecture – 26
Some Examples of Neural Networks (Contd.)

(Refer Slide Time: 00:15)



Now, we are going to solve one numerical example related to radial basis function network, and let us see, how can it model the input-output relationship of a process. Now, here, we are going to consider, in fact, a system having four inputs and one output only for simplicity. So, I am just going to show one radial basis network having, in fact, four inputs and one output.

(Refer Slide Time: 00:49)

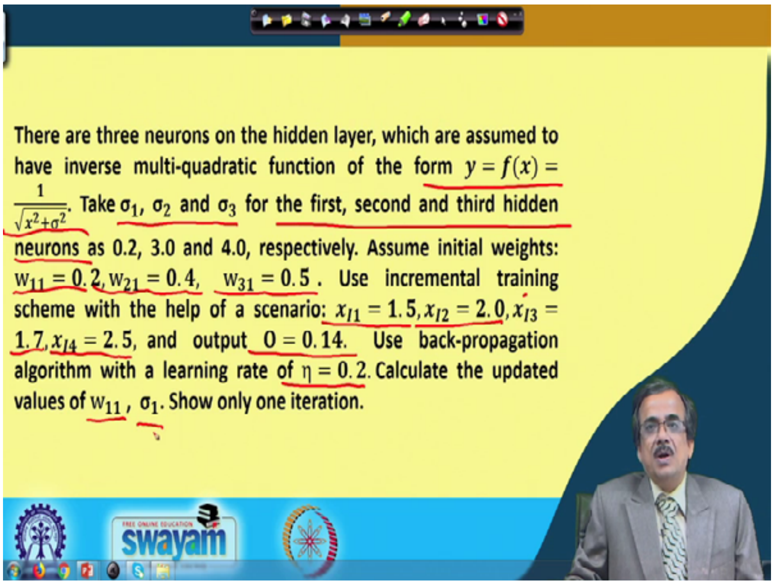


So, this is the network. Now, if you see, there are four input is here say x_I1, x_I2, x_I3, x_I4. And, this is nothing but actually the input node and we have got only one output. And here on the output layer we are using the linear transfer function. And, on the hidden layer, we are using a radial basis function network, say inverse multi-quadratic function. So, inverse multi-quadratic function I am using as the transfer function in the hidden layer and we have got three neurons on the hidden layer.

So, this H_I1 indicates the input of the first neuron lying in the hidden layer; H_O1 is the output of the first neuron lying on the hidden layer. H_I2 say input of the second neuron lying in the hidden layer; then H_I2 is the output of the second neuron lying on the hidden layer.

Then, H_I3 is input of the hidden neuron the third neuron lying on the hidden layer; and H_O3 is the output of the third neuron lying on the hidden layer. Now, these outputs are multiplied by the connecting weights and those are summed up here as the input and using the linear transfer function, we get this particular output. Let us see, how to carry out this particular analysis and how to solve the numerical example.

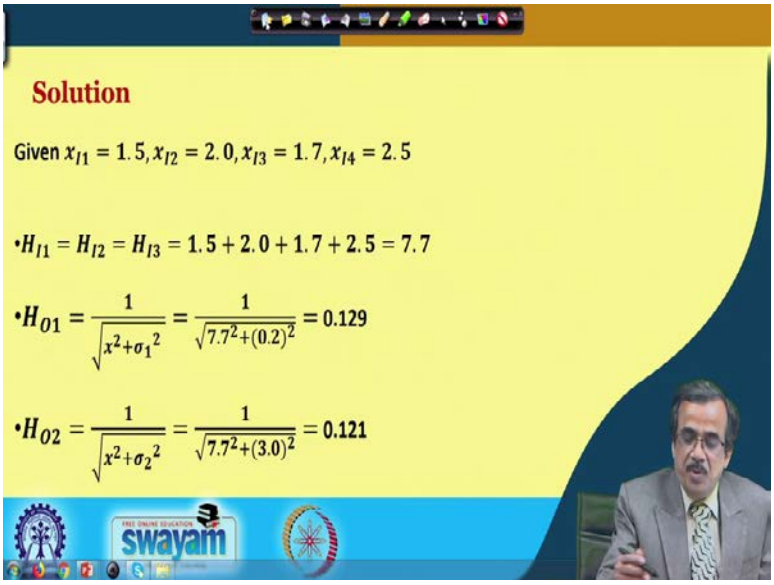
(Refer Slide Time: 02:43)



Now, let me give the statement. There are three neurons on the hidden layer, which are assumed to have inverse multi-quadratic function of the form, $y = f(x) = \frac{1}{\sqrt{x^2 + \sigma^2}}$, take σ_1, σ_2 and σ_3 for the first, second and third hidden neurons as 0.2, 3.0 and 4.0, respectively. Assume initial weights w_11 = 0.2, w_21 is 0.4, w_31 is 0.5.

We use incremental training scheme with the help of a training scenario: x_I1 is 1.5, x_I2 is 2.0, x_I3 is 1.7, and x_I4 is 2.5. The target output is nothing but 0.14. We are going to use back-propagation algorithm with a learning rate η equals to 0.2. And, we will have to update this w_11 and σ_1, and we are going to solve for one iteration.

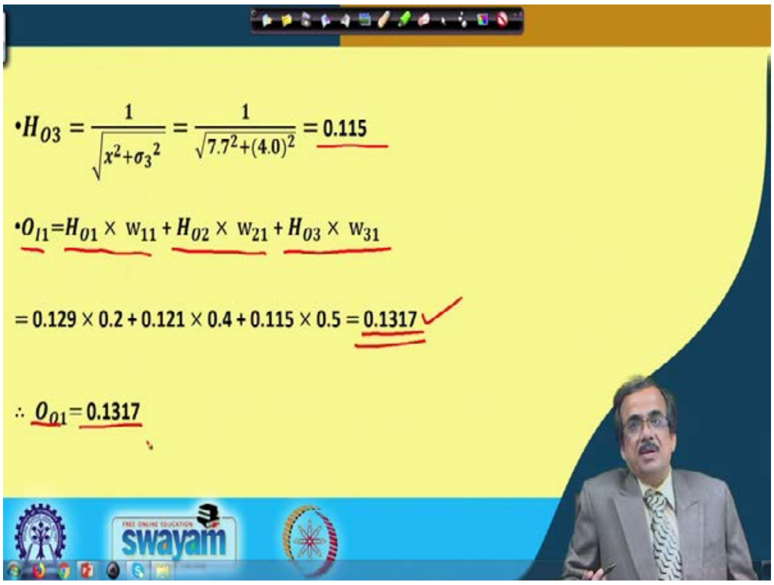
(Refer Slide Time: 04:15)



Now, let us see how to solve it. Now, here we have some given values like x_I1 is 1.5, x_I2 is 2.0, x_I3 is 1.7, and x_I4 is 2.5. Now, as I discussed, we try to find out H_I1, that is nothing but the input of the first neuron lying in the hidden layer and it is the same as input of the second layer lying on the hidden layer and it is same as input of the third neuron lying on the hidden layer is nothing but 1.5+2.0+1.7 +2.5 = 7.7.

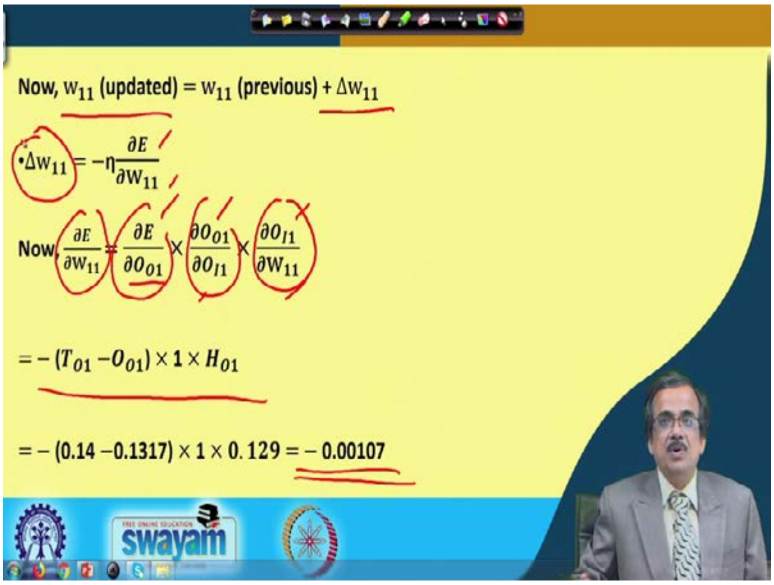
Now, this $H_{O1} = \frac{1}{\sqrt{x^2 + \sigma_1^2}}$. So, x is 7.7, σ_1 is 0.2. And, if you just insert these values and calculate, we are getting 0.129. Similarly, this $H_{O2} = \frac{1}{\sqrt{x^2 + \sigma_2^2}}$. If you substitute the values for x and σ_2, and if you calculate, you will be getting 0.121. So, this is the way actually we can find out H_O1, H_O2.

(Refer Slide Time: 05:49)



And, $H_{O3} = \frac{1}{\sqrt{x^2 + \sigma_3^2}}$; 1 divided by square root of 7.7 square plus 4.0 square that is nothing but 0.115. Then, we determine what should be the input of the first neuron lying on the output layer that is $O_{I1} = H_{O1} \times w_{11} + H_{O2} \times w_{21} + H_{03} \times w_{31}$. And, if you substitute all such values the numerical values, and if we calculate, you will be getting 0.1317. And, here, on the output layer, we are using the linear transfer function. So, the output of the neuron the first neuron laying one the output layer is nothing but its input and that is nothing but 0.1317. So, we will be getting this particular output.

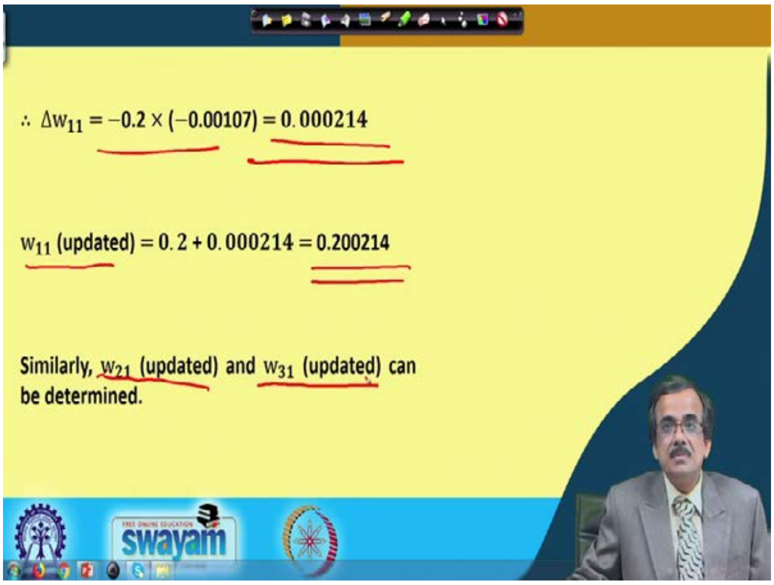
(Refer Slide Time: 07:07)



Now, based on this particular output, now we will have to find out this updated values, for the connecting weights and the update value for this particular the σ. Now, if you see by following the similar procedure, I can find out the updated value for this w_11 is nothing but w_11 previous plus delta w_11. Now, $\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}}$. Now, this particular partial derivative using the chain rule of differentiation we can write down, so 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 W_11 ok. And, now we can find out all such things like here this partial derivative that particular partial derivative and this particular partial derivative, we can find out.

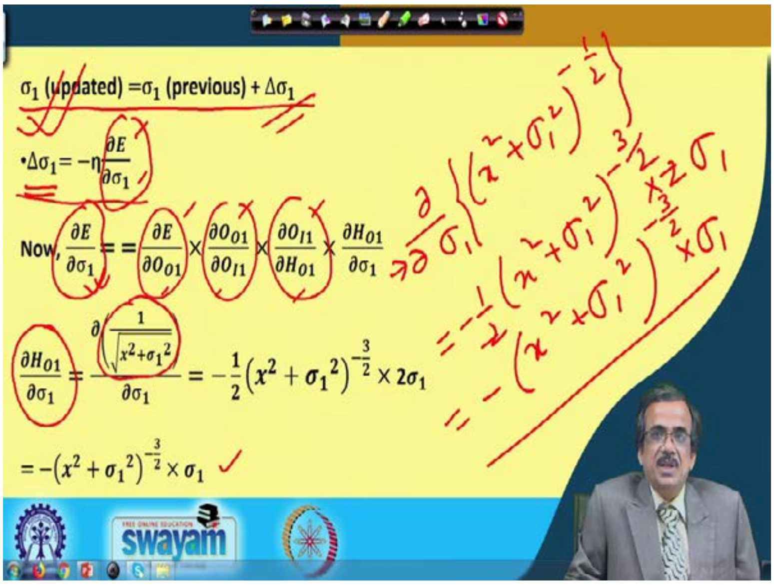
And, if you just substitute the numerical values, I will be getting partial derivative of E with respect to w_11 is nothing but this, and once you got this particular thing by multiply -η, so I will be getting this change in w_11.

(Refer Slide Time: 08:39)



Now, if you carry out this calculation, we will be getting your Δw_{11} is nothing but minus 0.2 (0.2 is the value of the learning rate) multiplied by minus 0.00107. And, if you just multiply, you will be getting this as Δw_{11}. Now, w_11 update is nothing but your the previous value plus the change in this. So, I can find out the updated value for this w_11. Now, by following the similar procedure, so I can find out the updated values for w_21, and the updated values for w_31.

(Refer Slide Time: 09:33)



And, once you have got this, so what you can do is, we can find out the updated value for this w_11 and other w's. Now, let us see, how to determine the updated value for this σ_1. Now, the updated value for σ_1 is nothing but the previous value for $\sigma_1 + \Delta\sigma_1$. Now, this $\Delta\sigma_1 = -\eta \frac{\partial E}{\partial \sigma_1}$. Now, partial derivative of E with respect to sigma_1 is nothing but partial derivative of E with respect to O_O1, partial derivative of O_O1 with respect to O_I1 multiplied by partial derivative of O_I1 with respect to H_O1 multiplied by partial derivative of H_O1 with respect to your σ_1.

Now, these particular derivatives very easily you can find out, this we have discussed several times. Now, let me concentrate on the last partial derivative that is partial derivative of H_O1 with respect to your σ_1, and how to determine this particular the partial derivative.

Now, it is very simple. Now, this can be written as the partial derivative of this, this is nothing but H_O1. So, this particular expression is your H_O1 with respect to σ_1. So, this is nothing but your if I just try to find out partial derivative or with respect to σ_1 of this particular expression, so this is nothing but your x square plus σ_1 square raise to the power your minus half.

So, if I just try to find out, how to find out, it is very simple. So, this is nothing but is your $-\frac{1}{2}(x^2+\sigma_1^2)^{-3/2} \times 2\sigma_1$; that means, you are. So, this 2, 2 gets cancelled. So, I will be getting $-(x^2+\sigma_1^2)^{-3/2} \times \sigma_1$. So, exactly the same thing which I have written it here, so very easily you can find out this particular the partial derivative.

And, once you got all such things very easily you can find out this particular partial derivative. And, once you have got all such things, very easily you can find out this partial derivative of E with respect to σ_1. And, once you have got it, I can find out what should be your the change in σ_1. And, once you got change in σ_1, we can find out your what is σ_1 updated.

Now, this is the way actually, we can update the connecting weights and this particular σ value. And, this process will go on and go on through a large number of iterations, and ultimately, you will be getting a network. And, this particular network will be able to make the prediction very accurately. Now, this is actually the working principle of the radial basis function network. Now, if I compare this particular radial basis function network with the multilayered feed forward network.

Now, in terms of accuracy like the both the networks are able to provide almost the same level of accuracy. But if I compare in terms of computational complexity, this radial basis function network is computationally faster compared to your multilayered feed forward network, and that is why, actually this radial basis function network has become very popular. And, this is very frequently used, in fact, to model input-output relationship of an adjunct process having say large number of inputs and outputs.

Thank you.