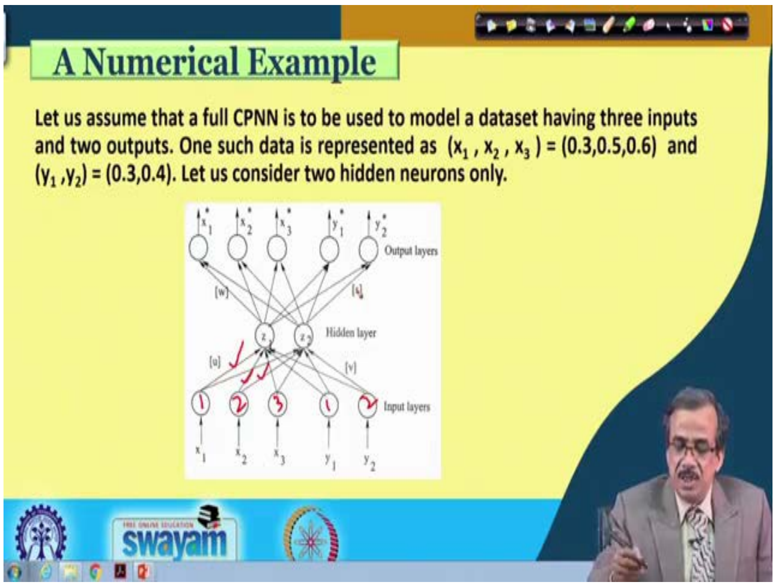
Fuzzy Logic and Neural Networks
Prof. Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 30
Some examples of neural networks (Contd.)

(Refer Slide Time: 00:15)



A Numerical Example

Let us assume that a full CPNN is to be used to model a dataset having three inputs and two outputs. One such data is represented as $(x_1, x_2, x_3) = (0.3, 0.5, 0.6)$ and $(y_1, y_2) = (0.3, 0.4)$. Let us consider two hidden neurons only.

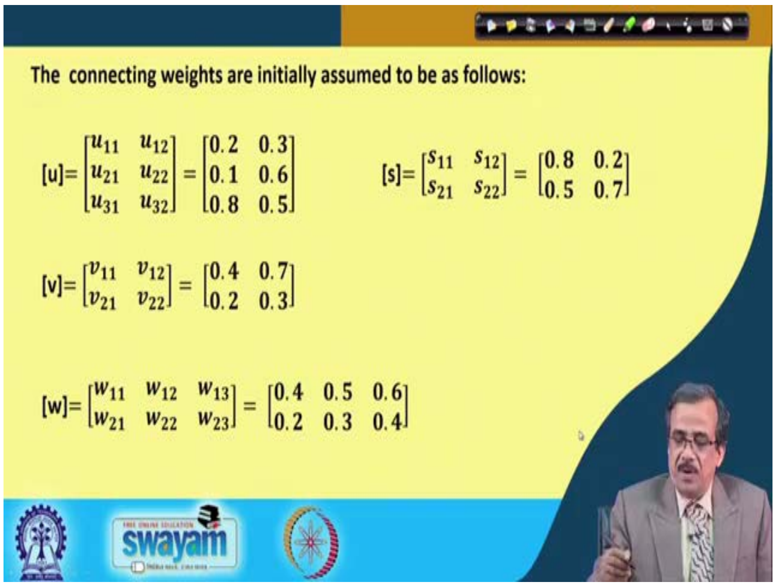
Now, we are going to solve a numerical example related to a full CPNN. Now, if you see the network, which I have already discussed, actually this is the schematic view of this full CPNN. And, let me give you the statement of this particular problem first, for this CPNN actually we are going to pass through 3 inputs and that is your x_1, x_2 and x_3.

Now, x_1, x_2 and x_3 are nothing but 0.3, 0.5 and 0.6 and its corresponding outputs are 0.3 and 0.4. And, let us consider that there are only two neurons in the hidden layer. So, we are considering two neurons in the hidden layer here and these x_1, x_2 and x_3 are the inputs and these are nothing but your the outputs. And, here, we consider two such input layers, one is this, another is this, and the connecting weights u, v, w, s, the way I discussed, and these are all star conditions for the x and star conditions for, the y.

Now, let us see the connecting weights. For these neurons, let me just put some numbers here and then it will be easy. So, let me consider say 1 here, 2 here, 3 here. So, this is nothing but u_11. So, this connecting weights is nothing but u_11, then this is nothing but u_21, then comes here, so this is nothing but u_22, and so on. Now, similarly here if

I just put 1 and 2, so the connecting weights between this and this, so this is nothing but is your v_11, then comes here v_12, and so on. Now, similarly, we have got the connecting weights: w and s here. Now, these numerical values of the connecting weights are assumed to be as follows:

(Refer Slide Time: 02:39)



The connecting weights are initially assumed to be as follows:

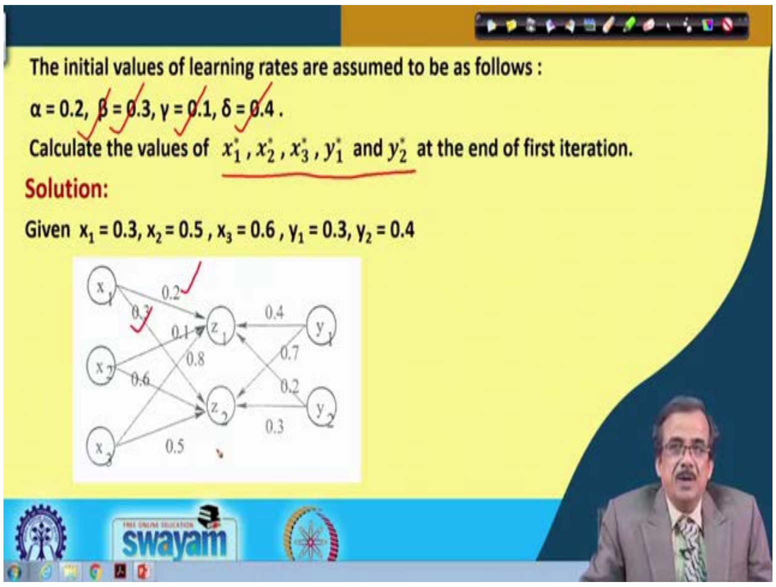
$$[u] = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.3 \\ 0.1 & 0.6 \\ 0.8 & 0.5 \end{bmatrix} \qquad [s] = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} = \begin{bmatrix} 0.8 & 0.2 \\ 0.5 & 0.7 \end{bmatrix}$$

$$[v] = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix} = \begin{bmatrix} 0.4 & 0.7 \\ 0.2 & 0.3 \end{bmatrix}$$

$$[w] = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = \begin{bmatrix} 0.4 & 0.5 & 0.6 \\ 0.2 & 0.3 & 0.4 \end{bmatrix}$$

Now, here, we have written all the initial connecting weights like u_11 is 0.2, u_12 is 0.3, and so on. So, this is the way actually we can write down the numerical values for the connecting weights. Then, v is nothing but this, then comes here s is nothing but this and your the w the connecting weight matrix is nothing but this.

(Refer Slide Time: 03:15)


The initial values of learning rates are assumed to be as follows:

$\alpha = 0.2$, $\beta = 0.3$, $\gamma = 0.1$, $\delta = 0.4$.

Calculate the values of $x_1^*, x_2^*, x_3^*, y_1^*$ and y_2^* at the end of first iteration.

Solution:

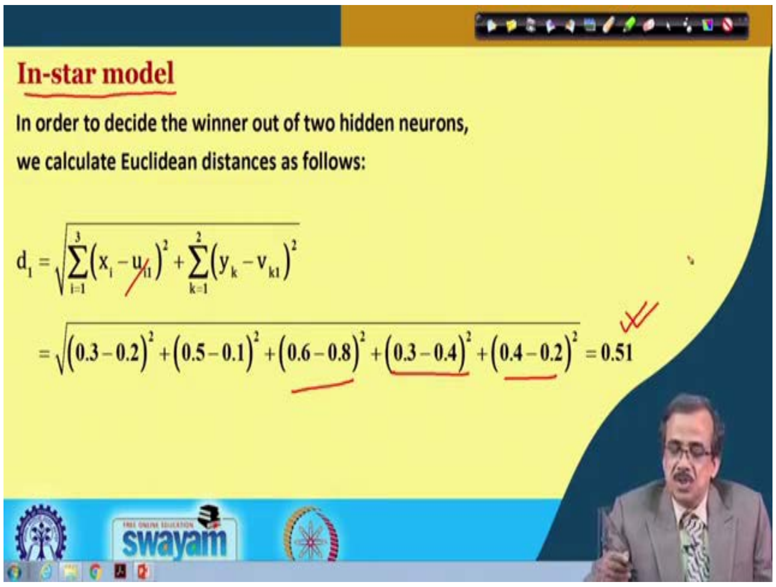
Given $x_1 = 0.3, x_2 = 0.5, x_3 = 0.6, y_1 = 0.3, y_2 = 0.4$

Now, if this is the situation, now, this shows actually the connecting weights and we can assume that the learning rate values are as follows: For example, say α is nothing but this, then comes β is 0.3, γ is 0.1, δ is 0.4 and our aim is to calculate like x_1^star, x_2^star, x_3^star, then y_1^star and y_2^star at the end of first iteration.

Now, here, we are going to solve only one iteration and let us see, how does it work. Now, this is another view of your this In-star model and these are nothing but the x inputs, these are the y inputs, so x_1 is 0.3, x_2 is 0.5, x_3 is 0.6 and y_1 is 0.3, and y_2 is nothing but 0.4. And, if you see the connecting weights, so these particular connecting weights are 0.2, 0.3, and so on and here also, we have got the connecting weights.

Now, the first thing, which we will have to do is, you will have to find the Euclidean distance from these two hidden neurons and we will have to declare a winner out of these two. Now, let us see, how to proceed with that particular calculation.

(Refer Slide Time: 04:43)



In-star model

In order to decide the winner out of two hidden neurons, we calculate Euclidean distances as follows:

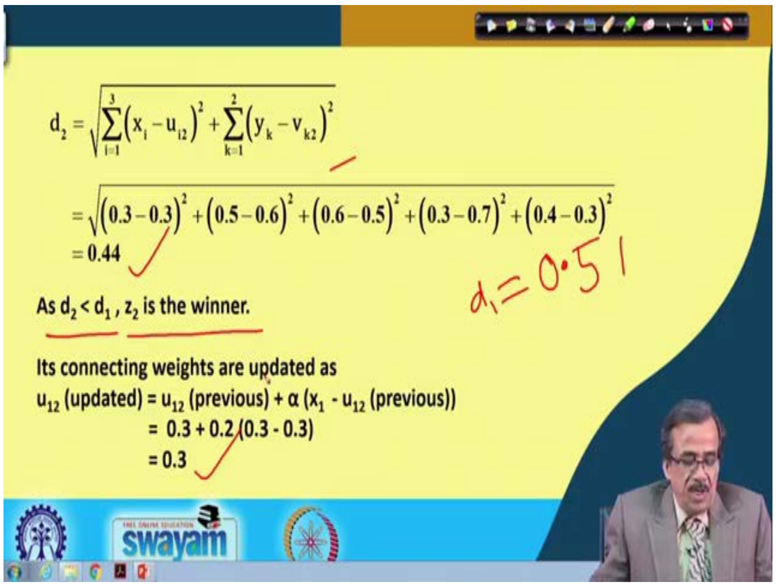
$$d_1 = \sqrt{\sum_{i=1}^{3}(x_i - u_{i1})^2 + \sum_{k=1}^{2}(y_k - v_{k1})^2}$$

$$= \sqrt{(0.3-0.2)^2 + (0.5-0.1)^2 + \underline{(0.6-0.8)}^2 + \underline{(0.3-0.4)}^2 + \underline{(0.4-0.2)}^2} = 0.51$$

That means, we are going to concentrate on your the In-star model. Now, here, in this in-star model, our aim is to find out this particular the distance value. Now, this $d_1 = \sqrt{\sum_{i=1}^{3}(x_i - u_{i1})^2 + \sum_{k=1}^{2}(y_k - v_{k1})^2}$.

Now, here, you can see that i varies from 1 to 3; that means, I put x_1 minus u_11 square plus x_2 minus u_21 square plus x_3 minus u_31 square. So, this is the thing. Next, we put k equals to 1 to 2. So, I put k equals to 1, so y_1 minus v_11 is nothing but this. Then k equals to 2, so y_2 minus v_21 is nothing but this and square. And, if we calculate, I will be getting this particular d_1 as 0.51. Now, on the hidden layer in fact, we have got two such neurons, so I will have to calculate, in fact, another d value. Now, that is your d_2.

(Refer Slide Time: 06:11)



$$d_2 = \sqrt{\sum_{i=1}^{3}(x_i - u_{i2})^2 + \sum_{k=1}^{2}(y_k - v_{k2})^2}$$

$$= \sqrt{(0.3-0.3)^2 + (0.5-0.6)^2 + (0.6-0.5)^2 + (0.3-0.7)^2 + (0.4-0.3)^2}$$

$= 0.44$ ✓

$d_1 = 0.51$

As $d_2 < d_1$, z_2 is the winner.

Its connecting weights are updated as

u_{12} (updated) = u_{12} (previous) + α (x_1 - u_{12} (previous))

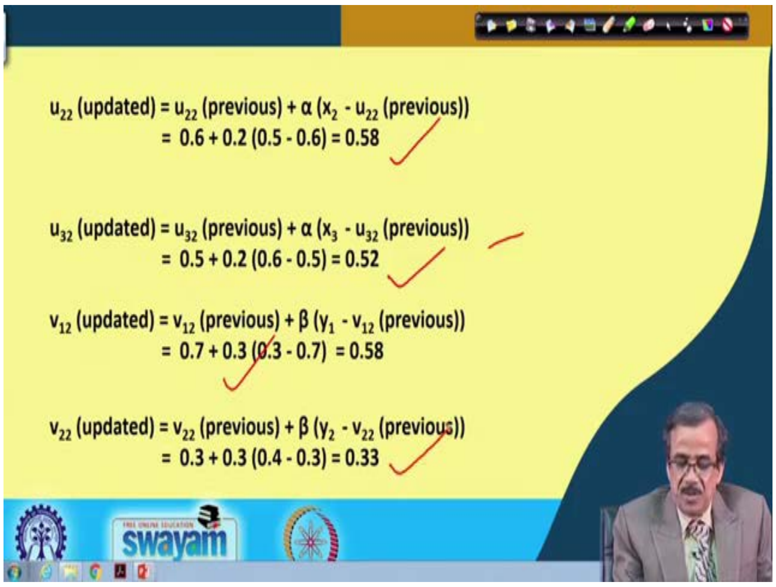
= 0.3 + 0.2 (0.3 - 0.3)

= 0.3 ✓

Now, this particular d_2 is nothing but square root summation i equals to 1 to 3 (x_i minus u_i2) square plus summation k equals to 1 to 2 (y_k minus v_k2) square. Now, I put i equals to 1, I will be getting (x_1 minus u_12) square plus (x_2 minus u_22) square plus (x_3 minus u_32) square plus summation k equals to 1 to 2. So, this is nothing but (y_k minus v_k2) square; k equals to 1, so (y_1 minus v_12) square plus (y_2 minus v_22) square and if you calculate you will be getting. So, d_2 is equal to 0.44.

Now, if you remember d_1 was your 0.51. And, if I compare this particular d_1 and d_2, so d_2 is found to be less compared to d_1. So, z_2 is actually the winner. So, through this competition, z_2 has been declared as winner. Now, once you got this particular winner. Now, we are in a position to update the connecting weights. For example, say $u_{12}(updated) = u_{12}(previous) + \alpha(x_1 - u_{12}(previous))$. Now, we substitute the numerical values and we can find out that this particular u_12 updated, it will be 0.3; that means, there is no change in this particular the u_12 value.

(Refer Slide Time: 08:14)



u_{22} (updated) = u_{22} (previous) + α (x_2 - u_{22} (previous))
= 0.6 + 0.2 (0.5 - 0.6) = 0.58

u_{32} (updated) = u_{32} (previous) + α (x_3 - u_{32} (previous))
= 0.5 + 0.2 (0.6 - 0.5) = 0.52

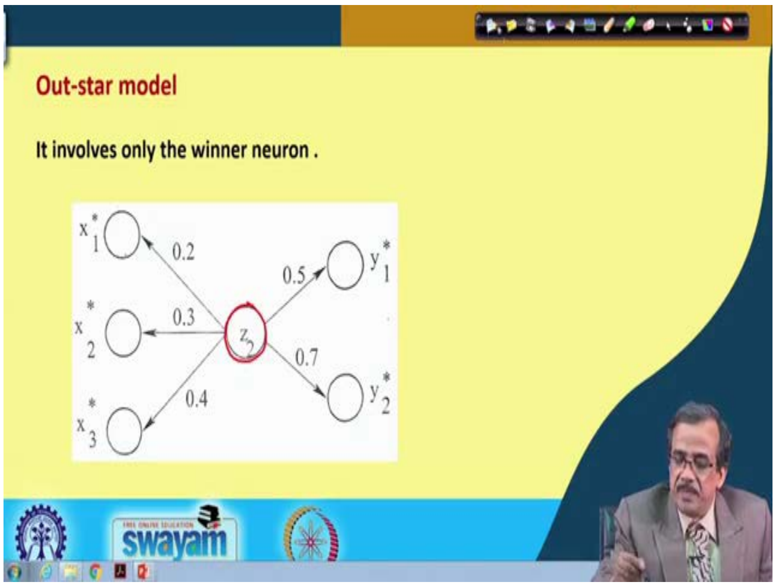
v_{12} (updated) = v_{12} (previous) + β (y_1 - v_{12} (previous))
= 0.7 + 0.3 (0.3 - 0.7) = 0.58

v_{22} (updated) = v_{22} (previous) + β (y_2 - v_{22} (previous))
= 0.3 + 0.3 (0.4 - 0.3) = 0.33

Now, here if you just go for the updating of the other parameters like your u_22 (updated) is nothing but u_22 (previous) plus α into (x_2 minus u_22) previous. Now, if you substitute the numerical values and calculate, so u_22 updated will be your 0.58, then comes here u_32 (updated) is nothing but u_32 (previous) plus α into (x_3 minus u_32) previous. And, if you substitute the numerical values you will be getting 0.52. Similarly, we can find out the updated values for your v, that is, v_12 (updated) is nothing but v_12 (previous) plus β multiplied by (y_1 minus v_12 (previous)). Once again, if I substitute the numerical values; so I will be getting say 0.58.

The next we will have to update is this v_22. So, v_22 (updated) is nothing but v_22 (previous), plus β into (y_2 minus v_22 (previous)), and if you submit the numerical values then you will be getting that v_22 updated is nothing but 0.3 3. Now, this the way actually, we can find out the updated value.

(Refer Slide Time: 09:44)

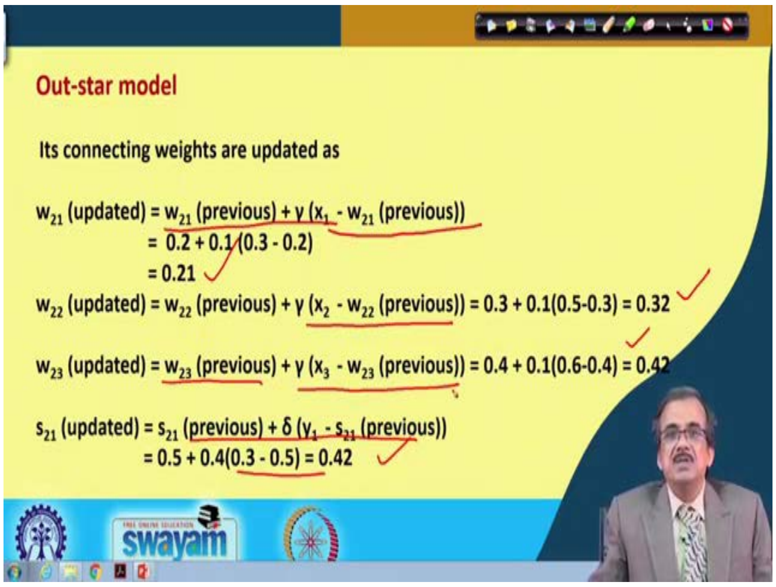


Out-star model

It involves only the winner neuron .

And, once you have got that. Now, we are in a position to carry out the Out-star model. Now, here in the Out-star model, so we will have to consider the winner hidden neuron and that is nothing but your z_2. And, if you see, this is one set of connecting weights, another set of connecting weights and those connecting weights once again we will have to update.

(Refer Slide Time: 10:14)



Out-star model

Its connecting weights are updated as

w_{21} (updated) = w_{21} (previous) + γ (x_1 - w_{21} (previous))
= 0.2 + 0.1(0.3 - 0.2)
= 0.21

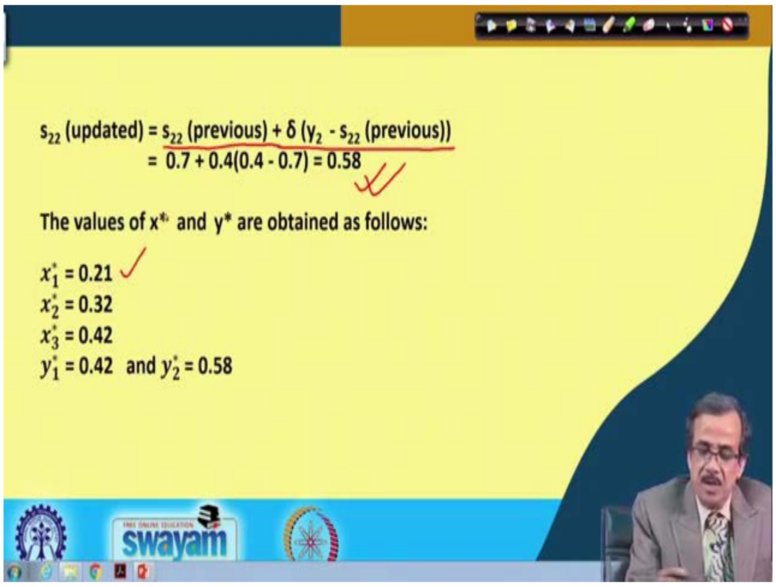
w_{22} (updated) = w_{22} (previous) + γ (x_2 - w_{22} (previous)) = 0.3 + 0.1(0.5-0.3) = 0.32

w_{23} (updated) = w_{23} (previous) + γ (x_3 - w_{23} (previous)) = 0.4 + 0.1(0.6-0.4) = 0.42

s_{21} (updated) = s_{21} (previous) + δ (y_1 - s_{21} (previous))
= 0.5 + 0.4(0.3 - 0.5) = 0.42

Now, if you update those connecting weights like your w_21 (updated) is nothing but w_21 (previous) plus γ multiplied by (x_1 minus w_21 (previous)). And if you substitute the numerical values and calculate, so w_21 updated will be 0.21. Similarly, w _22 (updated) it is nothing but w_22 (previous) plus γ multiplied by (x_2 minus w_22 (previous)) and if you substitute the numerical values you will be getting 0.32. Then, w_23 (updated) is nothing but w_23 (previous) plus γ into (x_3 minus w_23 (previous)) and if you substitute the numerical values you will be getting 0.42. Then, s_21 (updated) is nothing but s_21 (previous) plus δ into (y_1 minus s_21 (previous)), and if you substitute the numerical values you will be getting 0.42.

(Refer Slide Time: 11:30)



s_{22} (updated) = s_{22} (previous) + δ (y_2 - s_{22} (previous))
= 0.7 + 0.4(0.4 - 0.7) = 0.58

The values of x^* and y^* are obtained as follows:

$x_1^* = 0.21$ ✓
$x_2^* = 0.32$
$x_3^* = 0.42$
$y_1^* = 0.42$ and $y_2^* = 0.58$

Now, the same procedure, we will have to use for updating s_22. Now, s_22 (updated) is nothing but s_22 (previous) plus δ into (y_2 minus s_22 (previous)) and if you substitute the numerical values you will be getting that is equals to 0.58.

Now, what we do is, those updated values we are going to assign here. So, if you see x_1^star is nothing but 0.21. So, x_1^star is nothing but 0.21, x_2^star is 0.32, and x_3^star 0.42, y_1^star is 0.42 and y_2^star is 0.58. Now, this is the way actually, we can complete one iteration of these particular the CPNN.

(Refer Slide Time: 12:34)

References:

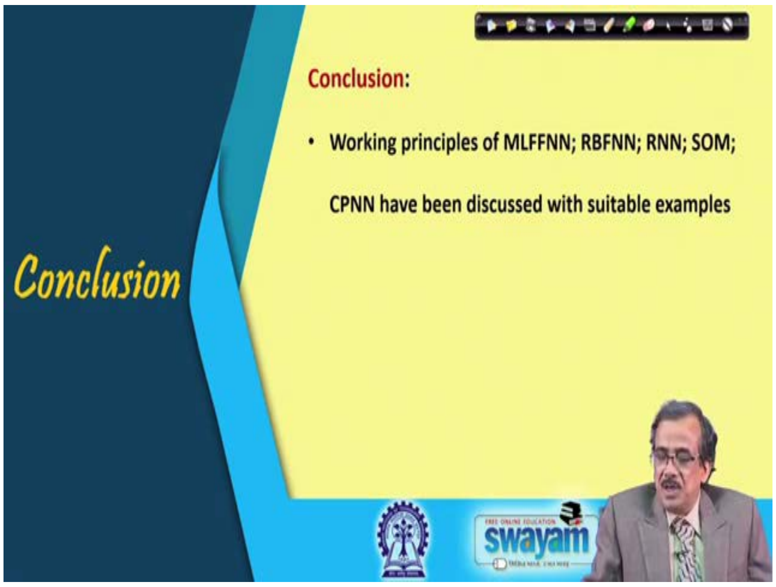
D.K. Pratihar Soft computing: fundamentals and applications. New Delhi, India: Narosa Publishing House, 2013.

Now, we will have to repeat and then, through a number of iterations, we will be getting that particular the relationship. Now, till now, we have discussed the working principle of multi-layered feed-forward network and that is very popularly known as the back-propagation neural network.

We have also discussed the principle of radial basis function network, then comes, we discussed recurrent neural network, and after that, we concentrated on the self organizing map, which works on unsupervised learning. And then, we concentrated on the counter propagation neural network, that is CPNN, which actually uses the concept of both supervised as well as your unsupervised learning. And, the main purpose of developing these particular networks is to establish the input-output relationship; so this problem is related to data mining, which we can solve using the different types of the neural networks.

Now, for your further study, you can concentrate on the textbook of this particular course, that is, Soft Computing: Fundamentals and Applications. So, you can see this particular textbook for more details.

(Refer Slide Time: 14:08)



Conclusion:

- Working principles of MLFFNN; RBFNN; RNN; SOM; CPNN have been discussed with suitable examples

So, let us summarize like which we have already discussed a little bit like, we have discussed the principle of multi-layer feed forward network, radial basis function network, then comes your recurrent network. So, these three networks works based on the supervised learning.

And, then, we concentrated on the self-organizing map, is a very efficient tool for dimensionality reduction or visualization. It is also an efficient tool for clustering and using the concept of self organizing map, we have discussed how to design and develop this counter propagation neural network. And, once again, the purpose is to model the input-output relationships.

Now, here, we have discussed different types of networks and let me repeat once again. The purpose of actually designing different types of networks is once again, how to model the human brain in the artificial way. Now, we have already discussed the principle of fuzzy logic, we have already discussed the principle of neural networks and we have seen like how to evolve the fuzzy reasoning tool, fuzzy clustering tool, and all such things.

And, in the next lecture, in fact, we are going to see how to evolve a particular network, and what is the basic principle, based on which we can evolve very efficient network, so that we can do this type of input-output modeling in a very efficient way.

Thank you.