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

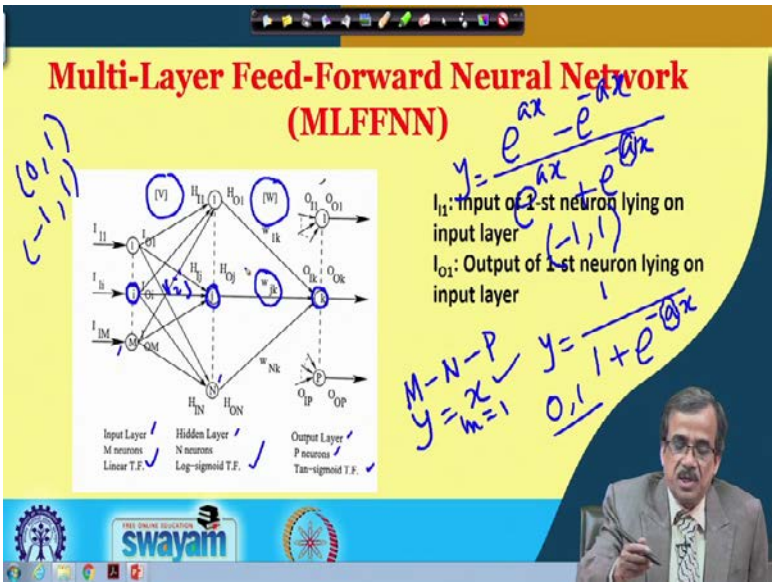
Lecture – 21
Some Examples of Neural Networks

Some Examples of Artificial Neural Networks: Now, we have seen how to design an artificial neuron by copying the principle of a biological neuron. We have also discussed, how to form a layer of neurons and we have also seen, how to design a particular structure of artificial neural network.

(Refer Slide Time: 00:53)

Now, we are going to discuss a some of the very important and popular neural networks. So, we are going to concentrate on the working principle of a few networks. Now, these networks are as follows: We are going to discuss the principle of multi-layer-feed-forward neural networks. Then, we are going to concentrate on radial basis function networks. After that, we are going to concentrate, how to include the feedback circuit along with the feed-forward circuit to develop the most popular, that is your recurrent neural networks. We will see the principle of self-organizing map or Kohonen network. And, at the end, we will try to see the working principle of a counter-propagation neural network and we will solve a few numerical examples also.

(Refer Slide Time: 01:57)



Now, here actually, let me start with the first one that is your multi-layer feed-forward network. And, in short, this is known as your MLFFNN, that is Multi-Layer Feed-Forward Neural Network. Now, its name indicates that this particular network consists of a number of layers. And, here, we are going to consider three layers, that is the input layer, then comes hidden layer and we have got the output layer.

Now, here on the input layer, we are going to consider capital M number of neurons. Similarly, on the hidden layer, we have got capital N number of neurons and on the output layer, we have got capital P number of neurons. So, this is known as in fact your M-N-P networks.

Now, here if you see on the input layer, we are using the linear transfer function say it is something like $y = x$ sort of thing, and we have considered the slope of this particular the straight line say m that is equals to 1. So, on the input layer, we are considering the linear transfer function $y = x$. On the hidden layer, we are considering the log sigmoid transfer function.

And, we have already seen the mathematical expression for the log sigmoid transfer function, that is nothing but $y = \frac{1}{1+e^{-ax}}$. Now, here, this particular a is actually the coefficient, which is going to decide, what should be the slope of this particular curve. Now, these are already mentioned that the higher the value of a, the steeper will be the

curve and vice-versa. So, this is your the log sigmoid transfer function and its output varies from your 0 to 1.

Now, on the output layer, we consider the tan sigmoid transfer function. And, if you see the mathematical expression for a tan sigmoid transfer function, this is something like this. $y = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}$. So, this is nothing but the tan sigmoid transfer function. And, once again, the coefficient a indicates the slope of the curve.

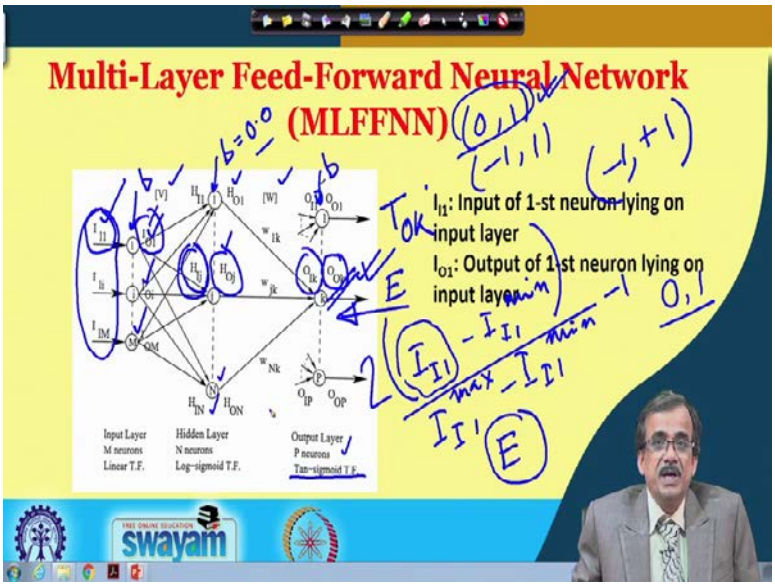
Now, if you see the range for this particular tan sigmoid transfer function, it varies in the range of – 1 to + 1. So, this is regarding the different types of the transfer function used in the different layers of this particular network.

Now, then comes here the connecting weights between the input layer and this particular your hidden layer is denoted by the V matrix. And, similarly, the connecting weights between the hidden layer and the output layer are denoted by so this particular the W matrix. And, if I see the individual values for this connecting weights, it may vary in a range of say 0 to 1 or in the range of say – 1 to + 1, that means, these are in the normalized scale.

Now, here if you see the inputs, as I mentioned there are M inputs and actually on the input layer, we have got the M number of neurons. Now, let me concentrate on a particular neuron lying on the input layer say ith neuron. And, a particular neuron that is your the j-th neuron lying in the hidden layer. And, a particular neuron lying on the output layer, say this is nothing but the k-th neuron. So, the connecting weight between i and j is denoted by your v_i j. And, the connecting weight between your j and k, this is nothing but is your w_j k.

Now, I am just going to discuss a little bit like how to design, in fact, your how to send the inputs to this particular network. Now, remember one thing so, the inputs to the neural networks are sent in the normalized form, because the different inputs may have different ranges.

(Refer Slide Time: 07:23)



And, if you do not normalized, there could be a possibility that this particular network may not work properly. So, what will have to do is, all the inputs like your so these inputs, we will have to pass in the normalized scale. Now, before that, if I how to make it normalized, so I am just going to discuss in details. Now, before that, let me just tell you the notations, which I am going to use here.

Now, this I _I1 this particular notation indicates, the input of the first neuron lying on the input layer, then comes here I _O1 is nothing but the output of the first neuron lying on the input layer. Similarly, if I see so this H _Ij is nothing but the input of the jth the neuron lying on the hidden layer. So, H_Oj is nothing but the output of the jth neuron lying on the hidden layer. And, this O_Ik is nothing but you are the input of the kth neuron lying on the output layer. And, this O_Ok is nothing but your output of the kth neuron lying on the output layer.

Now, I am going to discuss how to make this particular inputs in the normalized scale. Now, normalized scale means either we will have to put in the scale of say 0 to 1 or in the scale of say – 1 to + 1. So, let me discuss how to make it in the scale of say either 0 to 1 or from – 1 to +m 1.

Now, let me concentrate on this first one, that is I_I1. Now, to represent in the normalized scale, that is, 0 to 1 the formula, which is generally used is something like

this, that is, $\frac{I_{I1} - I_{I1}^{\min}}{I_{I1}^{\max} - I_{I1}^{\min}}$. Now, if input, that is, I_I1 is kept equal to your I_I1^minimum, I will be getting, which is nothing 0. And, if I put that I_I1 is nothing but I_I1^maximum, so I will be getting 1. So, this is the way actually, this particular input can be converted into the scale of 0 to 1.

Similarly, if I want to represent in the scale of – 1 to + 1. So, what I will have to do is actually, I will have to use one expression, which is nothing but $\frac{2(I_{I1} - I_{I1}^{\min})}{I_{I1}^{\max} - I_{I1}^{\min}} - 1$.

If I put I_I1 is equal to I_I1^minimum, so I will be getting 0 here that means I will be able to generate – 1. And, if I put I_I1 equals to I_I1^maximum, so here, I will be getting 2 -1. So, this is nothing but + 1. So, I can vary this particular input in the scale of – 1 to + 1.

Now, the point, which I am going to make that this particular inputs are to be represented in the form of normalized scale. And, once you got this particular inputs in the normalized scale, and if I use the linear transfer function on the input layer, I will be getting the output, that is nothing but equal to the input, that means, your I_Oi will become equals to I_Ii. So, this is the way, actually we can find out the output of the neurons lying on the input layer.

And, once you have got, this particular outputs of the first layer. Now, I can multiply with the corresponding connecting weight and I can sum them up, so that I can find out the input for the hidden layer. And, once you have got this particular input for the hidden layer, it will be passed through the log sigmoid transfer function. So, I will be getting the output like H_Oj, H_O1, and so on.

And, once you have got this particular outputs, now we will multiply with the corresponding connecting weights denoted by W and these are summed up here. And, that will be the input of the kth neuron lying on the output layer. And once again, it will pass to the transfer function, that is your tan sigmoid transfer function. And, accordingly, I will be getting the output of the kth neuron lying on the output layer.

So, this the way actually, we can carry out the forward calculation. And, we can find out, what should be the output for a set of inputs. Now, here before I go for the forward

calculation. I am just going to mention two things. One is your, we generally use some bias value, so we put some bias value but, here for simplicity actually, I have assumed that bias is equal to 0. And, this is actually, how to make this particular analysis a little bit simple.

Now, another thing I should mention that initially we generate all such connecting weight values like the V values and W values at random using the random number generator. And, then, through a large number of iterations, we try to find out what should be the updated values for these V and W, so that this particular network can make the prediction as accurately as possible.

Now, let me repeat once again for the set of inputs, if I know all the transfer functions at the connecting weights, I will be able to find out this particular output and this is nothing but the calculated output. Now, for this training scenario that means, for this set of input parameters, there is one known output. And that is nothing but the target output that means, if I write here, O_Ok is nothing but the calculated output of the kth neuron lying on the output layer, I can also write down.

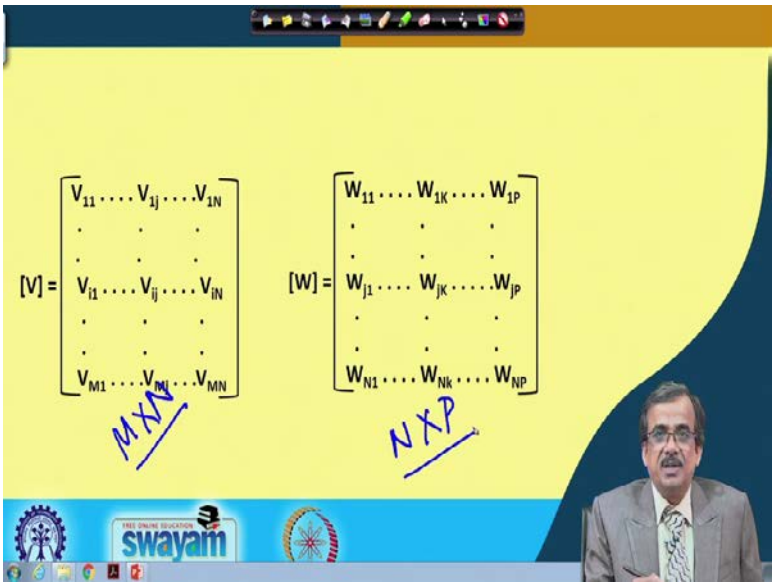
So, T_Ok is nothing but the target output of the k-th neuron. Now, if I know the target output, very easily I can find out the error. How to find out? I am just going to discuss in details. And, once I have got this particular error by comparing the calculated output with target output, I will propagate in the backward direction. I can modify or I can update all the connecting weights, and I can update the coefficients of the transfer functions. And, through a large number of iterations, I am just going to do this particular updating. And, ultimately, so this particular network is going to make the prediction as accurately as possible.

Now, another thing I should mention that the performance of this particular network depends on a number of parameters. For example, it depends on the connecting weights, it depends on your the coefficient of the transfer function, whether it is log sigmoid, tan sigmoid or whether it is the linear transfer function, it depends on the slope of the linear transfer function, it also depends on the topology or the architecture of this particular network.

And, to represent the topology or the architecture, we see how many layers are there and how many neurons are present in each of these particular layers. So, the performance of

this particular network not only depends on the architecture or the topology, it also depends on the connecting weights, the coefficient of transfer function, and all such things.

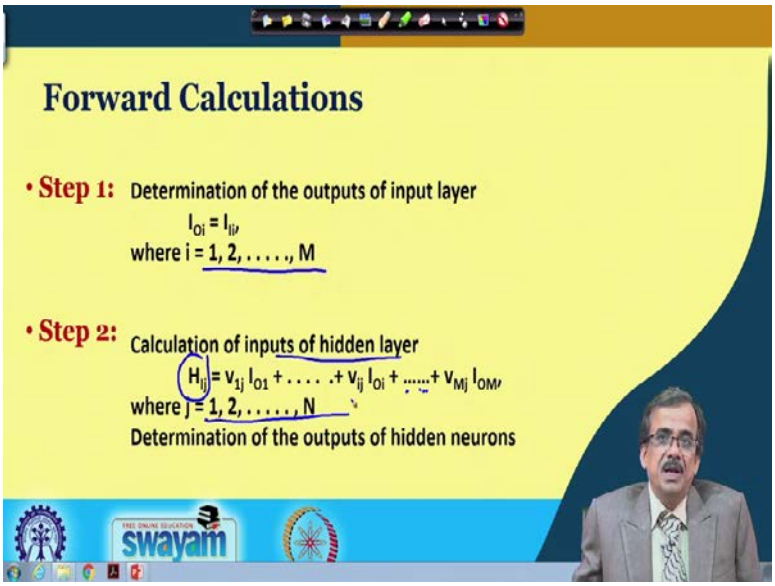
(Refer Slide Time: 16:17)



Now, let me see, how to carry out, actually, the forward calculation for this particular network. Now, before I go for that, this I have already mentioned that this V is nothing but the connecting weights between the input and hidden layers.

And, this is actually a matrix, the matrix of connecting weights. And, here, this is nothing but one $M \times N$. So, this is nothing but actually $M \times N$. And, this particular W matrix is a connecting weight matrix between the hidden and output layers. And, this is nothing but is your $N \times P$ matrix. Now, as I told that initially, this is generated at random, and through a large number of iterations, these particular connecting weights will be updated.

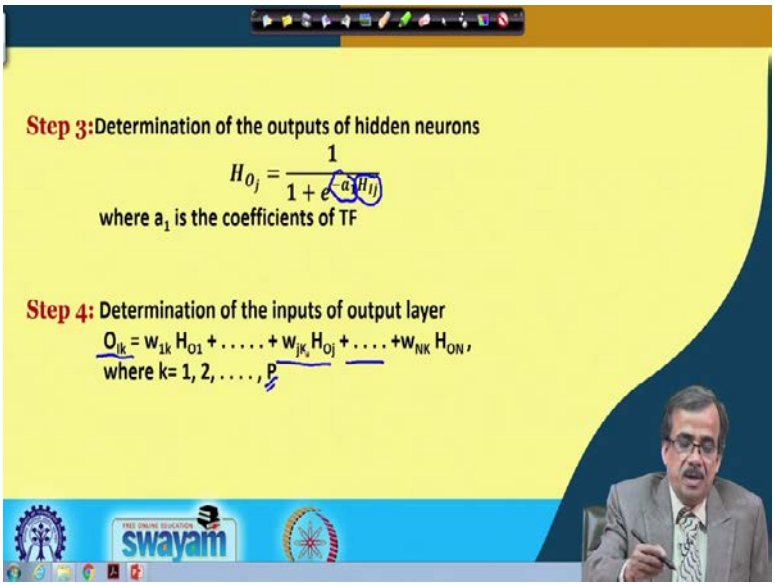
(Refer Slide Time: 17:05)



Now, let us see how to carry out the forward calculations. Now, in step-1, we try to determine the output of the input layer. Now, as we have considered the linear transfer function of the form $y = x$, so output equals to input. So, we can very easily write down, the output of the ith neuron lying on the input layer is nothing but the input of the ith neuron lying on the input layer. And, here, small m is varying from 1, 2 up to capital M.

Now, once we have got the output of the input layer, now we are in a position to determine what should be the inputs of the hidden layer. So, we are going to discuss step-2 and the input of the hidden layer that is nothing but $H_{Ij} = V_{ij} I_{O1} + \ldots\ldots\ldots\ldots + V_{ij} I_{Oi} + \ldots\ldots\ldots\ldots + V_{Mj} I_{OM}$. Now, here, j varies from 1, 2 up to N. And, once you have got the input of the hidden layer, now it will be passed through the transfer function just to find out, what should be the output of this hidden neuron.

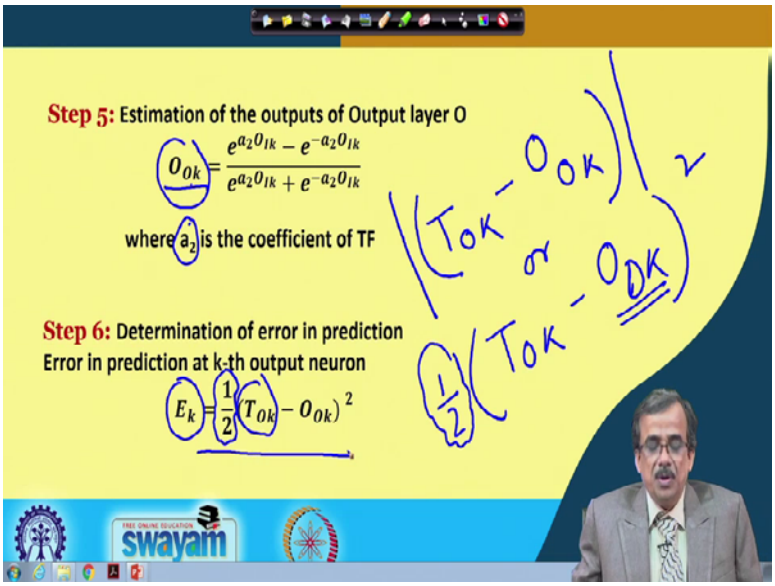
(Refer Slide Time: 18:49)



Now, let us see how to find out the output. So, step-3, that is, the determination of output of the hidden neuron, that is, $H_{Oj} = \frac{1}{1+e^{-a_1 H_{Ij}}}$. So, H_Ij is a actually nothing but the input of the jth neuron lying on the hidden layer. And, this particular a_1 is actually your the coefficient of the transfer function.

Now, next is your the step-4, that means, we will have to find out the inputs of the output layer. Now, this O_Ik is nothing but the input of the kth neuron lying on the output layer, and that is nothing but W_1k multiplied by H_O1 plus there are a few terms here plus W_jk multiplied by H_Oj plus there are a few terms here, and the last term is your W_Nk multiplied by H_ON, and k varies from 1, 2 up to capital P, and this P is nothing but the total number of neurons lying on the output layer.

(Refer Slide Time: 20:19)



Now, this is the way actually, you can find out the inputs of the output layer. And, once you have got the input of the output layer, now in step-5, we allow this particular input to pass through the transfer function, and you will be getting actually $O_{Oj} = \frac{e^{a_2 O_{Ik}} - e^{-a_2 O_{Ik}}}{e^{a_2 O_{Ik}} + e^{-a_2 O_{Ik}}}$.

And, this a_2 is nothing but the coefficient of the transfer function. So, this O_Ok is nothing but the calculated output of the kth neuron lying on the output layer.

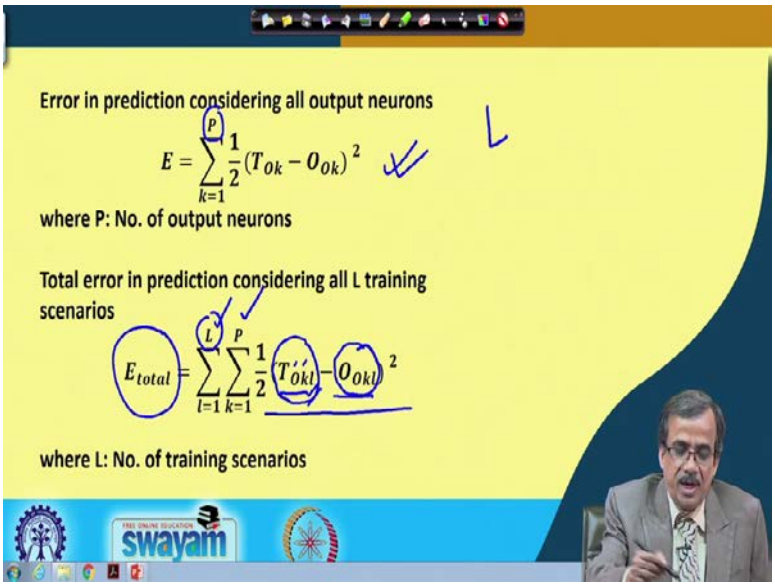
And, once you have got it, now we can compare with the target value, that is the output of the kth neuron that means your the target output that is denoted by T_Ok. Now, if I have got the target output is nothing but T_Ok and the calculated output is nothing but O_Ok, now this particular thing it could be either positive or negative.

And, that is why, to make it positive, what you do is, either we consider the mod value of this or what you considered is your like T_Ok minus O_Ok. So, square of that just to make it positive. And, here you can see, I have added here one term that is your half, so this is multiplied by half. The reason is actually very simple.

Now, in future, I will have to differentiate. So, this particular error of the kth output neuron with respect to the calculated one. And, if I put square here and if I differentiate, I will be getting one, 2. So, this 2 will be multiplied by 1/2 just to make it 1. For this

particular purpose actually, we use this particular term, that is, 1/2. So, we are able to find out what should of the $E_k = \frac{1}{2}(T_{Ok} - O_{Ok})^2$.

(Refer Slide Time: 22:45)



Now, once you have got this particular output of the kth neuron lying on the output layer. Now, in the output layer, in fact, we have got a large number of neurons or say P number of neuron. So, I can find out what should be the total error of all the neurons lying on the output layer and that is nothing but $E = \sum_{k=1}^{P} \frac{1}{2}(T_{Ok} - O_{Ok})^2$. So, this is nothing but the total output of the output layer considering P neurons.

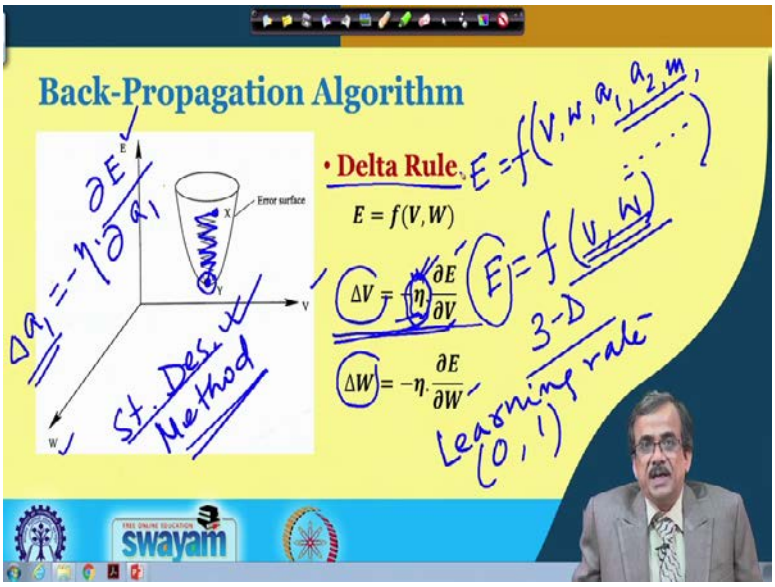
Now, if I have got this particular total error that corresponds to, in fact, only one training scenario, that means, only one set of inputs and outputs. And, supposing that we have got say capital L number of training scenarios, so what we will have to do is, we will have to pass all the training scenarios one after another and we can find out the total error after passing all capital L training scenario and that is denoted by E_total. So, $E_{total} = \sum_{l=1}^{L} \sum_{k=1}^{P} \frac{1}{2}(T_{Okl} - O_{Okl})^2$, where capital L indicates the number of training scenario.

Now, here, this particular term O_Okl represents the output of the k-th neuron lying on the output neuron layer corresponding to l-th training scenario. Similarly, say this

particular T_Okl indicates the target output of the k-th neuron lying on the output layer. And this particular output is the target output corresponding to the l-th training scenario.

So, this is the way actually, you can find out what should be the expression of the total error for this output layer neurons after considering all the training scenarios. And, once you got this particular picture, now, we are in a position like how to propagate it back, so that we can modify actually your the connecting weights and the coefficient of transfer function such that this particular network can make the prediction as accurately as possible.

(Refer Slide Time: 25:13)



Now, here actually, I am just going to discuss now how to minimize this particular error in prediction. Now, to minimize the error in prediction, what we will have to do is, we will have to take the help of one optimizer or one optimization algorithm. Now, here, I have already discussed that the performance of this particular network depends on a number of parameters. So, if I consider this error, error in prediction of the network, so that is a function of so many variables.

For example, say it depends on the connecting weights V, it depends on the connecting weights W, it depends on the coefficient of transfer function say a_1, it depends on the coefficient of transfer function a_2 for the output layer, it also depends on the slope of the linear transfer function, say m and there are a few other parameters. So, it depends on, in fact, a large number of parameters, ok.

Now, what you do is, for simplicity for the purpose of explaining how can it be minimized, how can this particular error be minimized. So, what we assume that this particular error is a function of only two variables, that is V, another is your W and for the time being you just forget the other terms. And, if I express that is error in prediction is a function of two variables V and W, so very easily, we can prepare the plot, that plot is nothing but the error plot. And, we will be getting this particular error surface.

Now, if I plot this particular error along the z direction and say, the connecting weight v along x direction and the connecting weight w along y direction, I will be getting this particular error surface in 3 D. Now, we know that we, human beings, can visualize only up to three dimension and that is why, we have considered that this particular error is a function of only two variables, so that we can plot this particular error surface in 3 D.

Now, if I consider more, so it will become four or more than four dimensions, which we cannot visualize. So, let us assume that this error is a function of only two variables, so on the 3 D plot, I can see, what should be this particular error surface. Now, as I told that we start with the random values for these particular the V and W and supposing that initially, the error of the network is here say.

And, what is our aim, our aim is to reach the minimum value of error, which is here. So, what you do is, we start from here, then through a large number of iterations, actually we try to move towards the minimum error solution and gradually, during the training, the network is going to reach this particular state, so that if we send a set of inputs, it will be able to predict the output as accurately as possible.

Now, as I told that we are going to take the help of some optimization tool for example, say we are going to use a very popular optimization tool, the traditional tool for optimization, that is known as the steepest descent method. So, the steepest descent method actually, we are going to use. Now, this steepest descent method is one of the most popular traditional tools for optimization. And, here, the search direction is opposite to the gradient.

So, we try to find out the gradient direction of the objective function and try to move in a direction opposite to the gradient. The same principle actually has been copied here in the back-propagation algorithm. Now, here, actually what you do is, we try to find out the change in V, that is the connecting weight between the input and the hidden layer,

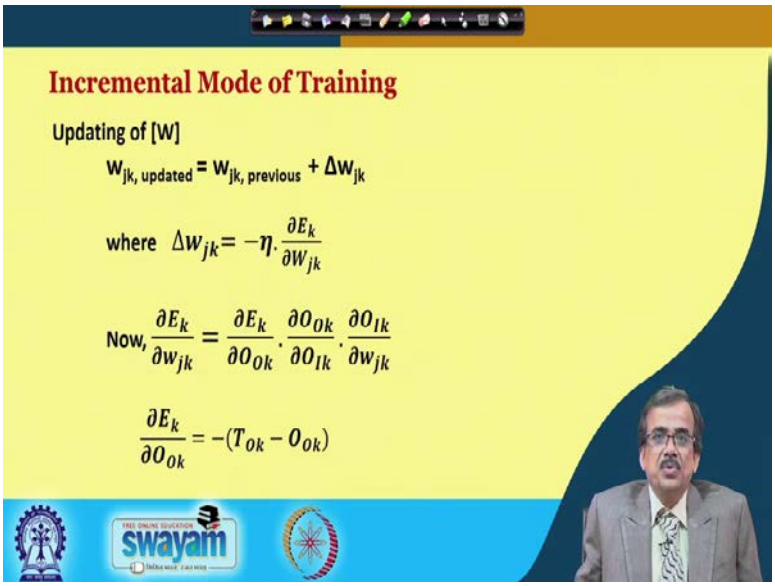
that is nothing but minus eta multiplied by the partial derivative of error E with respect to the V. $\Delta V = -\eta \frac{\partial E}{\partial V}$. Now, this particular actually the η is known as the learning rate. Now, this learning rate actually it varies in the range of, say 0 to 1.

Now, if I compare this particular expression with the expression of this particular, the steepest descent method, so the search direction is decided by $-\frac{\partial E}{\partial V}$, that means, I have moving in a direction opposite to the gradient. And, here so this η, that is the learning rate is going to represent actually the step length of this particular steepest descent algorithm. Now, let me mention that for any optimization tool, there are two things which will have to consider, one is your what should be the search direction, and what should be the step length.

Now, the same thing, we have just copied here to find out the change in V is nothing but eta (the step length) and minus partial derivative of E with respect to V, so that is going to decide, what should be the search direction. Now, similarly, actually the change in W is nothing but $-\eta \frac{\partial E}{\partial W}$. Similarly, if I just want to say find out the updated value for this the coefficient your a_1, a_2.

So, what will have to do is, so we will have to find out what is $\Delta a_1 = -\eta \frac{\partial E}{\partial a_1}$. And, similarly, I can also find out the small change in a_2 also. So, this is the way actually we can implement this delta rule just to find out what should be the updated values for the connecting weights and the coefficients.

(Refer Slide Time: 32:33)



Now, I will be discussing all such things in details mathematically we will see how to determine these updated values with the help of some mode of training and these things will be discussed, in details.

Thank you.