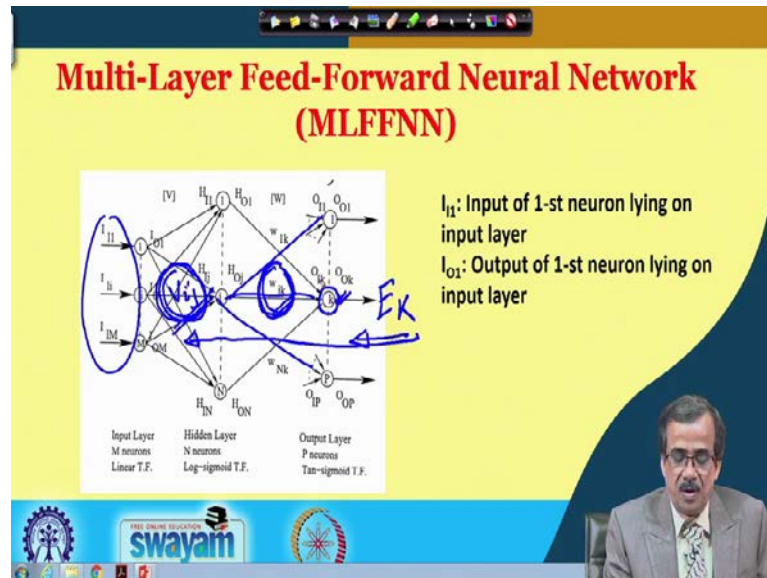


Fuzzy Logic and Neural Networks
Prof. Dilip Kumar Pratihari
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 22
Some Examples of Neural Networks (Contd.)

(Refer Slide Time: 00:17)



So, this is the multilayer feed forward network. I am supposing that we have passed say one set of training scenarios here, now if I pass one set of training scenarios and we have discussed like how to get this calculated output, and we have seen how to determine this particular error of the kth output neuron. Now, based on this particular error, this error has to be propagated back; so, I will have to update this particular w_{jk} and here, we have got the connecting weight say V_{ij} . Now, let us see how to update.

Now, to update that actually, what you do is, we try to move in the backward direction and the moment it reaches here we will stop. Now, here to update this V_{ij} , so starting from this particular error. I will propagate it back and the moment it reaches here, I am going to stop. And, another thing I am just going to tell you that this particular w_{jk} has got some contribution on the output of this particular kth neuron, but it has got no contribution towards the output of the other output neurons. So, w_{jk} has got contributions towards the output of this kth output neuron, that means, if I want to update this w_{jk} , I will have to consider only the kth output neuron and its error.

On the other hand, if I want to update this particular V_{ij} . So, depending on the V_{ij} , I will be getting some output here. Now, this particular H_{Oj} has got at least some contribution of this particular V_{ij} . And, here, this H_{Oj} is connected to all the output neurons, so whatever outputs we are getting at the different neurons of the output layer. So, this particular V_{ij} has point has got at least some contribution. That means, if I want to update this particular V_{ij} , I will have to consider the average error of this particular output neuron. On the other hand, if I want to update only w_{jk} , so I will have to consider the error of only this k th output neuron. So, I think, I am clear.

Now, with this particular understanding, so let me start with the principle of your the incremental training and let us see how to use the principle of incremental training.

(Refer Slide Time: 03:05)

Incremental Mode of Training

Updating of $[W]$

$$w_{jk, \text{updated}} = w_{jk, \text{previous}} + \Delta w_{jk}$$

where $\Delta w_{jk} = -\eta \frac{\partial E_k}{\partial w_{jk}}$

Now, $\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E_k}{\partial O_{ok}} \frac{\partial O_{ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial w_{jk}}$

$\frac{\partial E_k}{\partial O_{ok}} = -(T_{ok} - O_{ok})$

Handwritten notes:

Chain rule

$$E_k = \frac{1}{2} (T_{ok} - O_{ok})^2$$

$$\frac{\partial E_k}{\partial O_{ok}} = 2 \times \frac{1}{2} (-1) (T_{ok} - O_{ok})$$

A small video inset of a man in a suit is visible in the bottom right corner of the slide.

The principle of which, I have already discussed in details, now I am just going to discuss how to update this particular w_{jk} . Now, w_{jk} is nothing but the connecting weight between the j th hidden neuron and the k th output neuron. Now, this

$$w_{jk, \text{updated}} = w_{jk, \text{previous}} + \Delta w_{jk} . \text{ Now, this } \Delta w_{jk} = -\eta \frac{\partial E_k}{\partial w_{jk}} .$$

Now, let us see how to determine this particular partial differentiation, that is your

$$\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E}{\partial O_{ok}} \frac{\partial O_{ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial w_{jk}} . \text{ So, we are going to use actually the chain rule of}$$

differentiation. So, the chain rule of differentiation, we are going to use just to find out what should be this particular $\frac{\partial E_k}{\partial w_{jk}}$.

Now here, this $E_k = \frac{1}{2}(T_{Ok} - O_{Ok})^2$. Now, if I try to find out the partial derivative of this particular O_{Ok} , so I will be getting actually $\frac{\partial E_k}{\partial O_{Ok}}$ is nothing but your 2 multiplied by $1/2$ multiplied by minus 1 then comes your $T_{Ok} - O_{Ok}$. So, this 2, 2 gets cancel. So, this is nothing but $T_{Ok} - O_{Ok}$. So, this is the way actually, we can find out. So, this partial derivative is nothing but $-(T_{Ok} - O_{Ok})$. So, this is the way actually, this first partial derivative we can find out.

Now, we will have to find out the partial derivative of O_{Ok} with respect to your O_{Ik} and if you remember actually on the output layer, we use actually your the tan sigmoid transfer function.

(Refer Slide Time: 06:13)

Handwritten derivations on a yellow background:

$$\frac{\partial O_{Ok}}{\partial O_{Ik}} = a_2(1 + O_{Ok})(1 - O_{Ok})$$

$$\frac{\partial O_{Ik}}{\partial w_{jk}} = H_{Oj}$$

$$y = \frac{e^{a_2x} - e^{-a_2x}}{e^{a_2x} + e^{-a_2x}}$$

$$\frac{dy}{dx} = a_2(1+y)(1-y)$$

$$\frac{\partial E_k}{\partial w_{jk}} = -(T_{Ok} - O_{Ok})a_2(1 + O_{Ok})(1 - O_{Ok})H_{Oj}$$

$$\Delta w_{jk} = \eta a_2(T_{Ok} - O_{Ok})(1 + O_{Ok})(1 - O_{Ok})H_{Oj} \times H_{Oj}w_{jk}$$

Below the equations, it says $O_{Ik} = \dots$

Now, if you use tan sigmoid transfer function, so very easily, you can find out what should be the derivative. Now, let me concentrate on how to find out the derivative of this particular the tan sigmoid transfer function.

Now, if you just write down the expression like you have $y = \frac{e^{a_2 x} - e^{-a_2 x}}{e^{a_2 x} + e^{-a_2 x}}$. Now, if you find out the derivative, that is, $\frac{dy}{dx} = a_2(1+y)(1-y)$. So, we can find out this particular derivative and all of us you know, how to find out the derivative with respect to x and if you simplify, so you will be getting this particular the expression where y is nothing but this particular the expression.

The same thing, we have copied it here, just to find out $\frac{dO_{Ok}}{dO_{Ik}} = a_2(1+O_{Ok})(1-O_{Ok})$. So, very easily, you can find out this particular partial derivative. Now, then comes here the partial derivative of O_{Ik} with respect to your w_{jk} . Now, here, if you remember, so this O_{Ik} is what? That is nothing but the input of the kth neuron lying on the output layer, and if you just find out that particular expression that is your O_{Ik} there will be a few terms and in fact, there will be a few terms, but at the middle almost we will be getting a term that is nothing but H_{Oj} multiplied by your W_{jk} and there are a few other terms.

Now, what is this? This is nothing but the output of the jth neuron lying on the hidden layer multiplied by the connecting weight, that is your w_{jk} . Now, if I find out the partial derivative of O_{Ik} with respect to your w_{jk} . So, definitely I will be getting this particular H_{Oj} . So, very easily we can find out all the derivatives. And, once you have got all the derivatives, now you are in a position to determine what is $\frac{dE_k}{dw_{jk}}$. So, we substitute all the values, all the expressions, and then we will be getting this particular the final expression.

And, once you got this particular final expression of the partial derivative. So, we can find out Δw_{jk} is $-\eta$ multiplied by this particular partial derivative. So, I will be getting $\eta a_2 (T_{Ok} - O_{Ok})(1 + O_{Ok})(1 - O_{Ok})$ multiplied by H_{Oj} . So, very easily, we can find out what should be this particular change in w.

(Refer Slide Time: 09:41)

To update the connecting weight v_{ij} between i -th neuron of input layer and j -th neuron of hidden layer, that is,

$$v_{ij, \text{updated}} = v_{ij, \text{previous}} + \Delta v_{ij}$$

where $\Delta v_{ij} = -\eta \left\{ \frac{\partial E}{\partial v_{ij}} \right\}_{av}$

where $\left\{ \frac{\partial E}{\partial v_{ij}} \right\}_{av} = \frac{1}{P} \sum_{k=1}^P \frac{\partial E_k}{\partial v_{ij}}$

Now, once you have got the change in w . Now, we are going to discuss like your how to find out the change in V matrix. Now, let me concentrate on V_{ij} that is nothing but the connecting weight between the i th input neuron and j th hidden neuron. So, $V_{ij, \text{updated}} = V_{ij, \text{previous}} + \Delta V_{ij}$, and I have already discussed that this particular delta V_{ij} , if I want to determine.

So, we will have to consider the average effect, av stands for average. So, $V_{ij} = -\eta \left\{ \frac{\partial E}{\partial v_{ij}} \right\}_{av}$. Now, how to find out this particular expression? $\text{Del } E / \text{del } V_{ij}$ average is nothing but summation k equals to 1 to P . In fact, this would be your capital P , the notations which are using. So, $\left\{ \frac{\partial E}{\partial v_{ij}} \right\}_{av} = \frac{1}{P} \sum_{k=1}^P \frac{\partial E_k}{\partial v_{ij}}$.

(Refer Slide Time: 11:11)

Now,
$$\frac{\partial E_k}{\partial v_{ij}} = \frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial H_{Oj}} \frac{\partial H_{Oj}}{\partial H_{Ij}} \frac{\partial H_{Ij}}{\partial v_{ij}}$$

where,
$$\frac{\partial E_k}{\partial O_{Ok}} = -(T_{Ok} - O_{Ok})$$

$$\frac{\partial O_{Ok}}{\partial O_{Ik}} = a_2(1 + O_{Ok})(1 - O_{Ok})$$

Now, once you have got this particular expression, you will see how to determine, in fact, your this $\frac{\partial E_k}{\partial v_{ij}} = \frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial H_{Oj}} \frac{\partial H_{Oj}}{\partial H_{Ij}} \frac{\partial H_{Ij}}{\partial v_{ij}}$. Now, we have already discussed, how to determine, this partial derivatives, that is partial derivative of E_k with respect to O_{Ok} , that is nothing but this particular expression. We have also seen how to determine this particular expression of partial derivative and that is nothing but this particular expression.

(Refer Slide Time: 12:27)

$$\frac{\partial O_{Ik}}{\partial H_{Oj}} = w_{jk}$$

$$\frac{\partial H_{Oj}}{\partial H_{Ij}} = a_1 H_{Oj} (1 - H_{Oj})$$

$$\frac{\partial H_{Ij}}{\partial v_{ij}} = I_{Oi} = I_{Ii}$$

We get

$$\frac{\partial E_k}{\partial v_{ij}} = -a_1 a_2 (T_{Ok} - O_{Ok})(1 + O_{Ok})(1 - O_{Ok})(1 - H_{Oj}) w_{jk} H_{Oj} I_{Ii}$$

Handwritten note: $y = \frac{1}{1 + e^{-a_1 x}}$
 $\frac{dy}{dx} = a_1 y(1-y)$

And, now I am just going to discuss, how to find out the next one, that is, $\frac{\partial O_{jk}}{\partial H_{oj}} = w_{jk}$

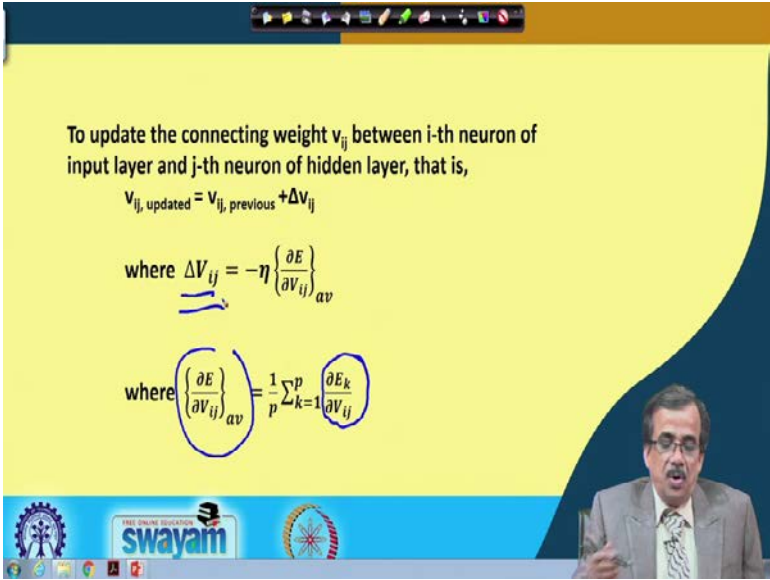
that also we have seen. Now, I am just going to find out the partial derivative of H_{oj} with respect your H_{ij} that means, output of j th neuron lying on the hidden layer and this is the input of j th neuron lying in the hidden layer.

Now, in the hidden layer actually, we have used one log sigmoid transfer function, which is nothing but $y = \frac{1}{1 + e^{-a_1 x}}$. And, if you find out its derivative that is $\frac{dy}{dx} = a_1 y(1 - y)$.

And, once you got this particular expression, very easily, you can find out the $\frac{\partial H_{oj}}{\partial H_{ij}} = a_1 H_{oj}(1 - H_{oj})$. And, the last term that is your $\frac{\partial H_{ij}}{\partial V_{ij}} = I_{oi} = I_{ii}$, that is output of

the i th neuron lying in the input layer and this is nothing but the input of the i th neuron lying in the input layer, so I can find out this particular expression.

(Refer Slide Time: 14:25)



To update the connecting weight v_{ij} between i -th neuron of input layer and j -th neuron of hidden layer, that is,

$$V_{ij, \text{ updated}} = V_{ij, \text{ previous}} + \Delta V_{ij}$$

where $\Delta V_{ij} = -\eta \left\{ \frac{\partial E}{\partial V_{ij}} \right\}_{av}$

where $\left\{ \frac{\partial E}{\partial V_{ij}} \right\}_{av} = \frac{1}{p} \sum_{k=1}^p \frac{\partial E_k}{\partial V_{ij}}$

And, once we have got it, now we can find out what is partial derivative of E_k with respect to V_{ij} and this is nothing but this particular the big expression. And, once you have got it, now we can also find out actually what should be your this particular V_{ij} . Now, V_{ij} is, in fact, nothing but your this particular expression. So, this is your this $\left\{ \frac{\partial E}{\partial V_{ij}} \right\}_{av}$ is nothing but this and we have already discussed how to determine this

particular thing and once you know this, I can find out this average and once I know this average I can multiply by η and put one negative side, that is nothing but is your change in V_{ij} .

So, we are in a position using this particular your differential calculus just to find out, what should be the change in connecting weights or the updated values for this particular the connecting weights using the incremental mode of training.

(Refer Slide Time: 15:25)

Batch Mode of Training:

Let us consider L training scenarios. Mean Squared Deviation in prediction for k-th output neuron

$$E' = \frac{1}{2} \times \frac{1}{L} \sum_{l=1}^L (T_{Ok l} - O_{Ok l})^2$$

The change in w_{jk} , that is, Δw_{jk} is determined as follows:

Now,

$$\Delta w_{jk} = -\eta \frac{\partial E'}{\partial w_{jk}}$$

$$\frac{\partial E'}{\partial w_{jk}} = \frac{\partial E'}{\partial E_l} \cdot \frac{\partial E_l}{\partial E_k} \cdot \frac{\partial E_k}{\partial O_{Ok}} \cdot \frac{\partial O_{Ok}}{\partial O_{Ik}} \cdot \frac{\partial O_{Ik}}{\partial w_{jk}}$$

Now, I am just going to discuss the batch mode of training. Now, this batch mode of training and its principle I have already discussed that supposing that I have got a large number of training scenario say capital L number of training scenarios. Now, if I got capital L number of training scenarios, what you do is, we pass all the training scenarios one after another and we try to find out, what is the error corresponding to the each of the training scenarios, we try to find out how much is the total error, we also try to calculate what is the average error and based on this particular average error, we update the network only once.

Now, let us see, how to implement this. Now, let us consider, there are capital L number of training scenarios. So, mean squared deviation in prediction for the k th output neuron that can be written as that is $E' = \frac{1}{2} \times \frac{1}{L} \sum_{l=1}^L (T_{Ok l} - O_{Ok l})^2$; and here actually, what will have do is, once again we will have to update the w_{jk} and v_{ij} , that is the connecting

weights between the hidden neurons and output neuron, and that between the input and the hidden neurons.

Now, let us see, how to update this particular w_{jk} . Now, this δw_{jk} is nothing but minus η multiplied by partial derivative of E^{prime} with respect to your w_{jk} . Now, how to determine this particular partial derivative? To determine the partial derivative, I am once again using the chain rule of differentiation. So, this $\frac{\partial E^{\text{prime}}}{\partial w_{jk}}$ is nothing but $\frac{\partial E^{\text{prime}}}{\partial E_l}$ that means, the error corresponding to the small l th training scenario then comes the rate of change of E_l with respect to your E_k then comes your the partial derivative of E_k with respect to O_{Ok} , partial derivative of O_{Ok} with respect to O_{Ik} , partial derivative of O_{Ik} with respect your w_{jk} .

Now, here I just want to mention one thing. Now, regarding the last three terms, for example, these three terms we can find out some numerical values, ok. So, ultimately you will be getting some numerical values. But, the first two terms, that means, your this particular partial derivative, it indicates only the rate of change of E^{prime} with respect to the l th training scenario, and, the rate of change of error with respect to the l th scenario, with respect to your the E_k (that is the error of the k th output neuron) are used just to tell you that we will have to sum all such things, but you may not get the direct numerical value corresponding to this first term and this particular the second term. But, starting from the third up to the fifth term, you will be getting some numerical values. Now, this is the way actually, we will have to find out the partial derivative of E^{prime} with respect your the w_{jk} .

(Refer Slide Time: 19:33)

Similarly, Δv_{ij} can be calculated as follows: $\Delta v_{ij} = -\eta \left\{ \frac{\partial E'}{\partial v_{ij}} \right\}_{av}$

Where $\left\{ \frac{\partial E'}{\partial v_{ij}} \right\}_{av} = \frac{1}{P} \sum_{k=1}^P \frac{\partial E'_k}{\partial v_{ij}}$

Now, $\frac{\partial E'_k}{\partial v_{ij}} = \frac{\partial E'_k}{\partial E_{kl}} \frac{\partial E_{kl}}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial H_{Oj}} \frac{\partial H_{Oj}}{\partial H_{Ij}} \frac{\partial H_{Ij}}{\partial v_{ij}}$

Now, then comes here, how to update the v_{ij} . Now, this Δv_{ij} is nothing but minus η the partial derivative of E^{prime} with respect to v_{ij} average. And, this particular $\frac{\partial E^{\text{prime}}}{\partial v_{ij}}$ average is nothing but summation k equals to 1 up to P , $\frac{\partial E_k^{\text{prime}}}{\partial v_{ij}}$ and will have to sum them up and then will have to multiplied by one divided by capital P . So, this is the way actually, we can find out the average of this particular partial derivative. And, how to find out this $\frac{\partial E_k^{\text{prime}}}{\partial v_{ij}}$. So, this

$$\frac{\partial E'_k}{\partial v_{ij}} = \frac{\partial E'_k}{\partial E_{kl}} \frac{\partial E_{kl}}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial H_{Oj}} \frac{\partial H_{Oj}}{\partial H_{Ij}} \frac{\partial H_{Ij}}{\partial v_{ij}}$$

Now, we have already discussed like how to find out these derivatives and I can also find out the numerical values corresponding to each of the partial derivatives. But, once again, the first, this particular component, that is the partial derivative of E_k^{prime} with respect to your E_{kl} , you may not be able to determine the numerical value, but it indicates the rate of change of E_k^{prime} with respect to E_{kl} . Similarly, this is nothing but your the rate of change of E_{kl} with respect your O_{Ok} . Now, these two terms is going to help us or going to tell us that you find out, this particular expression for each of the training scenarios and you sum them up, just to find out the total effect and, so that we can find out the average effect for updating of that particular the connecting weights.

Now, once you have got this particular derivative using this formula, I can find out this average and once I have got this particular average. So, I will be able to find out what

should be the change in v_{ij} . So, this is the way actually, we can update the connecting weights using actually the batch mode of training.

(Refer Slide Time: 22:25)

Momentum Constant (α')

Generalized Delta Rule:

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w}(t) + \alpha \Delta w(t-1)$$

η : (0.0 to 1.0)
 α' : (0.0 to 1.0) ✓

α' is used to ensure a stable network even at a higher value of learning rate η

Now, supposing that we know the updated values for the connecting weights. Now, if you see this particular rule, now let me just write it here, for example, say the change in w is nothing but (let me repeat) as $\Delta w = -\eta \frac{\partial E}{\partial w}$. Now, if I follow this particular rule that is called the delta rule. Now, there is a possibility that this particular partial derivative, it could be either positive or negative. But, this η is always positive and it is lying in the range of 0 to 1. Now, this η is positive, but this partial derivative could be either positive or negative. Now, supposing that it has become negative and here, I have got another negative side so that will make the Δw is actually a positive value in some of the iterations.

Now, that means, while updating this particular w , if I just go on adding some numerical values. So, after running this particular algorithm for a large number of iterations, there is a possibility that the value for the w may come out of the range. For example, the range is 0.0 to 1.0, another range could be minus 1.0 to plus 1.0. Now, if I just go on adding, so this particular updated value then what will happen actually is this weight may go out of the range and the network may lose the stability or the balance. Now, neural network actually does not know anything of the physical problem. So, if it is wrongly

trained, then also it is going to give some results, but we will have to be careful that the stability of this particular network should be maintained, that means, your w should not exceed its own range.

Now, supposing that w is found to be greater than 1, in that case, we will have to use some correction and that correction is something like this. So, what I will have to do is, if w is found to be greater than 1. So, I am just going to put 1 divided by w . Now, if I consider 1 divided by w , this will become less than 1 and once again, I can change this particular network for more iterations. The moment I put 1 divided by w in place of w , there could be a sudden change in the performance of the network, but after a few iterations, once again it is going to reach that particular balanced region and this particular network can be believed, only it is working in that particular balanced region.

Now, to overcome this particular problem, so that w does not become greater than 1, so what we do is, we use the generalized delta rule and that is nothing but is your $\Delta w(t)$, (t indicates t -th iteration) is nothing but minus η multiplied by the partial derivative of E with respect to w corresponding to the t -th iteration plus α' multiplied by $\Delta w(t-1)$. So, this particular extra term we are going to add and α' is known as the momentum constant and the range for the momentum constant is once again from 0 to 1, and what we do is, we try to see: what was the change of w in the previous iteration that is nothing but $(t-1)$ -th iteration. That means, we try to see the history of this particular the updated weight that means, what happens at $(t-1)$ -th iteration to this particular Δw , that I am going to consider.

Now, if I consider then there is a chance that I am going to provide some sort of damping effect to this particular network or I am just going to put some sort of cushioning effect to this particular network, so that the w does not exceed its range, but if I follow this once again there is no guarantee that the connecting weight will lie within its range, if the network is running for a large number of iterations, so it may once again go out of this particular range. And, if it goes out of the range, this is actually the remedy.

Now, as I told, just to put some sort of damping effect to this particular network, we consider its history that means, what happens to the previous iteration, we want to give some weightage to this particular change in connecting weight, just to find out, what

should be your the updated weight. So, this is the way, we can update, this particular the network.

(Refer Slide Time: 27:57)

Notes:

- BP algorithm → there is a chance of local minima problem
- BP algorithm → transfer functions are to be differentiable in nature
- BPNN may not be able to capture the dynamics of a highly dynamic process
- Inputs are normalized ✓
- Connecting weights lie in the range of either (0.0, 1.0) or (-1.0, 1.0) ✓
- Convergence Criterion: Absolute value of the rate of change of error in prediction becomes less than or equal to a pre-specified value
- A neural network can be either fully-connected or partially-connected one

The slide also features a diagram of a neural network with three layers of nodes and a small video inset of a man speaking in the bottom right corner.

Now, a few notes here I have put, I just want to mention. Now, this multilayer feed forward network trained with the help of this back-propagation algorithm, that is the delta rule or the generalized delta rule, there is a possibility that there will be a local minima problem. The reason is very simple because it works based on the steepest descent algorithm and that means, it is going to use the information of the gradient.

Now, supposing that say I have a very complicated error function with so many such ups and downs and undulations. Now, what you do is, while minimizing the error, this particular algorithm will try to find out the search direction, which is opposite to the gradient. Now, gradient is a local property. So, there is a chance that it is going to get stuck at the local minima, and it will not be able to reach that particular your globally optimal solution and this back propagation algorithm is actually having a chance of local minima. And, the transfer function has to be differentiable because we are using the gradient information, and at the point of discontinuity, in fact, we will not be able to find out the gradient of this particular objective function or the error function.

Now, this back propagation neural network or multilayer feed forward network trained using the principle of back propagation algorithm, may not be able to capture the dynamics of a highly complex or highly dynamic process and that is why, actually we

will have to go for some feedback circuit, which I will be discussing in details, that means, will have to go for the recurrent network. Those things will be discussed in much more details, while discussing the recurrent networks.

Inputs are to be normalized I have already mentioned the range for the connecting weights have already mentioned, and actually, we will have to find out some convergence criteria for this particular the algorithm during the training. Now, if the rate of change of error in prediction becomes less than or equal to some pre-specified small value, we say that the network has reached that particular optimal situation and we consider that particular network is an optimal network.

Now, a neural network could be either a fully-connected network or it could be a partially-connected network. Now, very quickly, let me take a very simple example. So, if I have got say one network having this type of structure. So, it is 2 input neuron, 3 hidden neuron and 1 output neuron, and if the connectivity is something like this, the all neurons are connected. So, this type of network is known as the fully connected network. On the other hand, if I have got a network of this type like, if I have got a network of this type, say I have got these are the input neurons, and supposing the these are the hidden neuron and this is output neuron.

(Refer Slide Time: 31:29)

Notes:

- BP algorithm → there is a chance of local minima problem
- BP algorithm → transfer functions are to be differentiable in nature
- BPNN may not be able to capture the dynamics of a highly dynamic process
- Inputs are normalized
- Connecting weights lie in the range of either (0.0, 1.0) or (-1.0, 1.0)
- Convergence Criterion: Absolute value of the rate of change of error in prediction becomes less than or equal to a pre-specified value
- A neural network can be either fully-connected or partially-connected one

The diagram shows a neural network with 2 input neurons, 3 hidden neurons, and 1 output neuron. Arrows indicate connections between the input and hidden layers, and between the hidden and output layers. Some connections are solid, while others are dashed, illustrating a partially-connected network.

swayam

And, if the connectivity is something like this, so this is nothing but actually one partially connected network, because this connectivity is actually not ensured. Then comes, your

this connectivity is not ensured. So, dotted lines are actually absent. So, this type of network is known as the partially-connected network.

Thank you.