Course Name: Machine Learning and Deep learning - Fundamentals and Applications

Professor Name: Prof. M. K. Bhuyan

Department Name: Electronics and Electrical Engineering

Institute Name: Indian Institute of Technology, Guwahati

Week-11

Lecture-38

Welcome to NPTEL online course on machine learning and deep learning fundamentals and applications. In my last class, I explained the concept of the deep learning and also I explained the concept of the convolutional neural network that is CNN. For the training of the network, I can employ the back propagation training algorithm that already I have explained. And if I increase the number of layers of a particular network, it is expected that the performance of the network may improve. Also the network can extract more number of features from the input image. That means, I will be getting more representations of the input data or the input image with the help of more number of layers.

So I am repeating this. So if I increase the number of layers of a particular convolutional neural network or the deep architecture, it is expected that the performance may improve. But practically it is not happening in many cases because of the problem of the vanishing gradient. The opposite of the vanishing gradient is the exploding gradient.

So today I am going to discuss this concept, the concept of the vanishing gradient. So before explaining this concept, I will be explaining the concept of the single layer perceptron. And after this, I will be explaining the concept of the multilayer perceptron and also the back propagation training algorithm. And after this, I will be explaining the concept of the vanishing gradient problem. So let me begin this class.

So in this case, I have shown one single layer perceptron SLP. So we have the input signals X1, X2 up to Xm, you can see in the figure. So I have the input signals. And you can see I have shown the summing junction. And you can see the connecting weights.

So W_{k1}, W_{k2}, W_{km}, these are the weights, the connecting weights between the input node and the summing junction. And also we are considering the bias input. B_k is the bias

input. And you can see these are the input signals. So this is the single layer perceptron.

So we do not have any hidden layers, we have the input layer. And you can see the output, that is the output node. And we are considering the activation function. So at the kth node, the input Xj, these all the inputs are multiplied by their weights, the corresponding weights along with the bias B_k are sum. So already this concept I have explained.

So I have to consider weighted inputs. So input is multiplied with the corresponding weights. And we have to sum up. And also we have to sum the bias. So we have to consider bias also.

At kth node, the inputs Xj multiplied by their weights, the weights are W_{kj} along with the bias B_k are sum. And we are considering the activation function phi that controls the magnitude of the accumulated output. So why I need the activation function that already I have explained in my artificial neural network class. So this activation function phi controls the magnitude of the accumulated output. So mathematically how to determine Yk.

So Yk is the output. So that is nothing but $\phi(Vk)$. So Vk is nothing but the accumulated output. So that is nothing but first what we are considering the inputs are multiplied with the corresponding weights plus bias, the bias is the B_k. So it is sum up.

So j is equal to 1 to m because we are considering m number of input signals X1, X2 up to Xm. So this is the mathematical representation. And in the vector from I can write in this from that is nothing but $W_k^T X + B_k$ and B_k is the bias input. So this is the concept of the single layer perceptron. So in this case we are considering this activation function ϕ is the activation function and Vk is the accumulated input.

So the output of this is 1 if you see the output, output is 1 when the Vk > 0. That is the $W_k^T X + B_k > 0$ that is nothing but 1 the output will be 1. Otherwise the output will be 0. So for this we are considering the step function, step function as an activation function. So here I have shown the step function as an activation function and the bias is considered to change the position of the decision boundary.

But I cannot sense the orientation of the decision boundary only I can sense the position of the decision boundary based on the bias value. So bias changes the position of the decision boundary. So this is the concept of the SLP the single layer perceptron. Now during the training what will happen you can see because already I told you SLP is good only for the simple classification problems. And if the problem is very simple and suppose the problem is the linearly separable classes then we can employ the single layer

perceptron.

So here you can see I am considering the inputs x1, d1. So corresponding to x1 input the desired output is d1 and corresponding to xn the desired output is dn. So because this is the supervised training so corresponding to particular input what is the desired output that I know. So corresponding to x1 the output is d1 that is the desired output corresponding to x2 the desired output is d2 like this I have the training samples. So during the training first I have to randomly initialize the weights and after this we have to compute y that y I can compute that is the output we can compute.

So that is nothing but W(t). xj so we are having this equation the input is multiplied with the weights and we are considering the activation function the activation function is ϕ. After this we have to update the weights the weight W(t + 1) that is the weight in the t + 1 iteration is equal to W(t) that is the weight in the tth iteration plus we are considering the error the error is nothing but the desired output minus actual output that is considered that is the error. Because already I told you during the back propagation training the error is back propagated to the input. So this dj - yj that is nothing but the difference between the desired output and the actual output that we are considering and we have to minimize the error. So for this I am updating the weights by considering this weight updation rule.

So this is the weight updation rule we are considering. So I have to do the iterations until the cost function and the cost function already I have defined that is nothing but the difference between the desired output and the actual output. So until a particular convergence condition is not obtained. So you can see the error should be less than the parameter the parameter is γ. So γ is the user defined parameter that is the threshold and if the cost function is less than this γ then I can stop the iteration.

So by considering this back propagation training we can find a decision boundary between the classes but the problem is it can be employed if the problem is very simple the problem means the pattern classification problem and the classes should be linearly separable then only I can apply the single layer perceptron algorithm. So for complex tasks I have to consider multi-layer perceptron MLP that is the feed forward network with hidden layers. In SLP I do not have hidden layers only input layer and output layers are available but in case of the MLP I have to consider hidden layers between input layer and output layer. So the drawbacks of the SLP so the output of a perceptron can take only one of two binary value so either it is 1 or 0. So it is not also this SLP not able to solve complex problems the complex pattern classification problems and picks up many solutions of varying quality.

So that already I have explained in my previous slide so it is very difficult to find the best decision boundary between the classes. If the vectors are not linearly separable learning will never reach a point where all the vectors are correctly classified. So if the problem is not linearly separable then this SLP cannot give good accuracy and the problem is the misclassification. So if I consider a very complex problem and if the samples are not linearly separable then I have to consider MLP the multi-layer perceptron. So let us move to the next slide.

So in this figure I have shown the concept of the multi-layer perceptron MLP. So input is first test from using a learned nonlinear transformation. So maybe we can employ some nonlinear transformations for mapping that concept I have explained in case of a radial basis function. So this allows nonlinearly separable input data to become linearly separable. So I have to consider nonlinear transformations of the input data and if I do this transformation and the data will be linearly separable.

So that concept I have explained in case of the radial basis function and in the figure I have shown one single hidden layer MLP. So you can see the input layer. So this is the input layer. So the nodes are X1, X2, X3 and we are considering one hidden layer and the nodes are H1 and H2 and we have the output node that is Y. So it is a very simple network that is a single hidden layer MLP we are considering.

So what will be the output at the node Y? So we are considering the activation function ϕ. Now I need to determine the output of the node Y. So by considering this equation that is nothing but we are considering the activation function ϕ and weight is multiplied with the input. The input is Hj and the corresponding connecting weight is Wj. So I am getting the output Y that is nothing but the ϕ that is the activation function.

The Wj is multiplied with the input the input is Hj. So this is my input. So this expression the $\phi(\sum_j WjHj)$ that can be represented like this $\phi(\sum_j Wj\, \phi(\sum_i WijXi))$ because what is the output? What is the output in the node H1 and H2? So that is nothing but the input Xi is multiplied with the corresponding weights the weights are Wij. So corresponding to Hj that is the output my input is Xi. So Xi is multiplied with the corresponding weights that is the connecting weights Wij and we are considering the activation function ϕ.

So I can represent like this. Now I have to optimize the loss function J(W). So that is nothing but the minimum and we are considering the difference between the actual output and the desired output. So d_K is the desired output so that the loss function we are considering. So by considering this equation we can determine the actual output and we are considering the desired output that is the d_K is the desired output.

So we can consider this loss function. So the d_K is the target to the output node Yk. So d_K is the target to the output node Yk. So in this case the k is equal to 1 because we are considering only one node in the output layer. So we can consider the back propagation training for weight updation.

So move to the next slide. So what is the back propagation algorithm? So it has two phases first one is the propagation and another one is the weight updation. So in the propagation forward propagation of a training patterns input through the neural network in order to generate the network outputs value. So we are getting the output values corresponding to the input patterns the input values and after this the back propagated of the output activations through the neural network using the training pattern target in order to generate the difference between the target and the actual output value. So that concept already I have explained. So that is the cost function that is nothing but the difference between the actual output and the target value that is the desired output.

So that we have to minimize for minimization of this I have to adjust the weights of the artificial neural network. So we have to calculate the gradient loss function. So gradient value we have to calculate because I have to find a minimum value. So that is why I am taking the gradient of this. And finally we are considering the weight updation rule.

So that is nothing but $W(t + 1) = W(t) - \gamma \nabla(J(W))$ and we are considering the gradient of the loss function. So we are updating the weights and this gamma controls the speed and the quality of the learning. So that means it controls the learning rate. So that is the parameter gamma is the user defined parameter and it controls the speed and quality of the learning. So in this discussion I am considering the sigmoid activation function.

So activation function is $\sigma(x) = \frac{1}{1+e^{-x}}$. So one advantage of the sigmoid function is that the derivative can be easily computed for the sigmoid function. So here you can see I am just computing the derivative of the sigmoid function. The sigmoid function is $\frac{1}{1+e^{-x}}$ and you can see just I am taking the differentiation and finally I am getting this expression $\sigma'(x)$ that is the first order derivative is equal to $\sigma(x)(1 - \sigma(x))$. So you can see it is easy to determine the derivative of the sigmoid function.

So that is why in artificial neural network we consider the sigmoid function. So during the weight updation already I told you we considered gradient descent algorithm. So that algorithm already I have explained in one of my classes. So you can see in y axis we are considering the cost function and W is the weight and this point we have to reach that is the global cost minima. So that point I have to obtain during the iteration that is the global minimum.

So suppose my initial point already I have shown this is my initial weight and if I take the gradient of this then the gradient at this point will be in this direction in this direction that will be the direction of the gradient. But to get the minimum I have to move in the opposite direction of the gradient. So that is why this minus sign is because of this. So I am moving in the negative direction of the gradient. So that is why we considered the negative sign and this γ is nothing but the learning rate.

So if the learning rate is very high then during this iteration so this value may go to this point this value may go to this point like this I have the oscillations and for the convergence it will take time. But if I consider this γ is small then you can see this convergence the convergence will be like this. So I am getting this one like this I am getting these gradients and after this I am getting this minimum point that is a global minimum point I am obtaining. If I consider the γ is very high then the faster will be the convergence but it may not be accurate. But if I consider lower γ then I will be getting the more accurate convergence during the training but it will take time.

So that is the concept of the gradient descent algorithm. So I am moving in the opposite direction of the gradient and the parameter γ controls the learning rate. The parameter γ controls the learning rate. If γ is high then the faster is the convergence. If γ is low then the more accurate in the training but it will take time for the convergence.

Now I will show the mathematical derivation of the back propagation learning algorithm. So here I am considering a multi-layer perceptron that is the feed forward network. I have one input layer one hidden layer and one output layer. So corresponding to this input layer the inputs are X1, X2, X3, X4 these are the nodes of the input layer and H1, H2, H3 these are the nodes of the hidden layer and I have two nodes in the output layer. The connecting weights between the input layer and the hidden layer that is W_{ij} and the connecting nodes between the hidden layer and the output layer is W_{jk}.

So a single hidden layer ANN is considered for simplicity. So let us consider the derivation of the back propagation training algorithm. So we can determine Yk. So we are considering the σ. σ is the activation function.

The weight W_{jk} is multiplied with the input. So input is H1, H2, H3 for the output Y1, Y2. So now we are considering the weight updation for the hidden and the output layer. So only this portion of the network we are considering. So inputs will be H1, H2, H3 and the output will be Y1 and Y2.

So what is the Yk? Yk is nothing but $\sigma(\sum_j WjkHj)$. I am multiplying the inputs. The input is H1, H2, H3 with the weights, the corresponding weights W_{jk}. So this expression

I am obtaining by replacing Hj by this one. So what is Hj? Hj is nothing but the output corresponding to the input.

The inputs are X1, X2, X3, X4. So in terms of this I can write Hj. So Hj is nothing but σ if you see this part σ and we are considering the input is multiplied with the connecting weights and it is sum up. That means we are considering the weighted inputs. And the cost function is again that is J(W) that is nothing but the desired output and the actual output. So we are considering Yk is the actual output and Dk is the desired output that is the target output.

And based on this I am formulating the cost function. So now I have to take the derivative of the cost function with respect to the weight and in this case we have to do for two cases one is for the output nodes and one is for the hidden nodes. So we have to do the differentiation with respect to the weight and in this case I have to consider the output nodes and the hidden nodes. So you can see here in the output node I am differentiating the cost function with respect to the connecting weights.

So the connecting weights are W_{jk}. So this is the connecting weights. So that can be written like this $\frac{\partial J(W)}{\partial in_k}\frac{\partial in_k}{\partial Wjk}$. So what is in_k? That is the input to the output node Yk. So in_k is the input to the output node Yk. So this $\frac{\partial}{\partial in_k}\frac{1}{2}\sum_{k=1}^{K}(\sigma(in_k)-d_k)^2$. So this expression I am getting that is nothing but the J(W) that is the cost function and this part I am differentiating in_k with respect to W_{jk}.

So what is in_k? Ink is nothing but W_{jk} Hj. So that is the in_k. So in_k we can determine and that is nothing but W_{jk} Hj and if I take the differentiation of this one so this will be this. So I am just taking the differentiation of this. This is the sigmoid function and finally I am getting this expression. The expression is δ_kHj. So what is the δ_k? δ_k is nothing but $(\sigma(in_k)-d_k)\sigma'(in_k)$.

So $\sigma'$is nothing but the first order derivative. So this expression you can see. It is a very simple expression and these mathematical steps you have to see how to derive this equation. Now at the hidden node we have to consider this one. The differentiation of the cost function with respect to the weight W_{ij}. So W_{ij} is nothing but this W_{ij} that is the connecting weights between the input nodes and the hidden nodes.

So this I can write in this form. So that is $\frac{\partial J(W)}{\partial in_j}\frac{\partial in_j}{\partial Wij}$. So that is by the chain rule we are considering this expression. So that is by considering the chain rule I am getting this expression. So this J(W) already I have defined that is nothing but the difference between the actual output and the desired output. And this is nothing but what is in_j? in_j is nothing but it is the input to the hidden node.

So that is nothing but $W_{ij}Xi$. So the input Xi is multiplied with the corresponding weights and it is sum up. So that is the in_j and after this I have to do the mathematics. You can see the steps all the steps. So just I am doing the differentiation and finally I am getting the expression that is δ_jXi and what is δ_j? The δ_j is I am obtaining like this. So it is a very simple derivation and in this derivation you have to see how to get this expression.

So at the hidden node I am determining this and in the previous slide at the output node we are determining this. So at the output node we are determining the differentiation of the cost function with respect to the connecting weights and at the hidden node again I am determining the differentiation of the cost function with respect to the connecting weight. So connecting weight is W_{ij}. So I am getting this expression. So now how actually we do the back propagation training so that illustration I want to show you.

So this is a simple network I am considering and the inputs are X1 and X2. In this network I have 2 hidden layers. So this is my input. So this is my input and I have 2 hidden layers and this is my output node. So how to do the back propagation training in this case? So let us see. So you can see first I am determining Y1 that Y1 I can determine that is the output Y1 I can determine corresponding to this node.

So Y1 is nothing but F1 is the activation function $W_{(X1)1}$ that is the connecting weight between X1 and the node of the hidden layer. So $W_{(X1)1}$ X1 that is the X1 is the input $W_{(X2)1}$ so that $W_{(X2)1}$ you can see the width the connecting weight between the X2 and the node the node is 1 the hidden node is 1. So $W_{(X2)1}$ X2 so we are getting the value Y1 that is the output in the hidden node 1 of the first hidden layer. So after this we are determining the activation this activation we are determining so that is nothing but Y2 Y2 is nothing but the activation function F2 and we are just multiplying the weight $W_{(X1)2}$ you can see in the figure into the input the input is X1 and $W_{(X2)2}$ X2. So it is a simple equation and we are determining the activation in the hidden neuron.

We are determining the activation in the hidden neuron and similarly the Y3 corresponding to this neuron can be determined like this. So we are determining the activation or the output corresponding to this hidden neuron and after this we are considering the second hidden layer and you can see this activation or this output we can determine like this F4 that is the that is the activation function W14.Y1 + W24.Y2 + W34.Y3 so that we are determining and after this we have to consider the second the node this node we are considering and we are getting the output output is Y5 we are getting. After this come to the last node of the output node so that is the output node and we are determining the activation that is the output Y is equal to F6 that is the activation F6 (W46.Y4 + W56.Y5) so that we are considering to get the activation at the output

node. Now let us consider how actually we do the back propagation so we are determining the difference between Z-Y so Y is nothing but that is the actual output we have determined and Z is suppose the desired output that is the target output we are considering. So δ is nothing but the error that is the difference between the target value and the actual value so Z is the target value and Y is the actual value so that difference we are considering that is the error.

So error should be back propagated to the input to adjust the weights of the artificial neural network so you can see in the previous slide we are determining δ so this error is back propagated you can see so δ_4 we are determining that is nothing but W46.δ so we can now determine the δ_4. So corresponding to this hidden node we can determine the δ_4 and corresponding to this node hidden node we can determine δ_5 so δ_5 is nothing but W56.δ so you can see I am back propagating the error after this come to the next hidden layer so if I consider this is the hidden layer the corresponding this hidden node I can determine δ_1 that is nothing but W14.δ_4 + W15.δ_5 so I am moving towards left that means I am moving towards the initial layers similarly corresponding to this node this δ_2 is determined and finally I am adjusting the weights the connecting weights between the input node and the hidden nodes that is the $W_{(X1)1}$ so this is the updated width and $W'_{(X2)1}$ so I have two weights that is the corresponding to this node corresponding to this hidden node I have two weights connecting weights so one is $W_{(X1)1}$ and another one is $W_{(X2)1}$ so these two weights I am updating by considering the width updation rule so by considering the width updation rule I am updating the weights the connecting weights between the input node and the hidden nodes so I am obtaining $W'_{(X1)1}$ and $W'_{(X2)1}$ I am obtaining similarly I am getting the values $W'_{(X1)2}$ and $W'_{(X2)2}$ so all these connecting weights I am determining by this width updation rule so that is the width updation rule for the back propagation training and in this case you can see I am just back propagating my error from the end to the beginning from the end to the beginning I am back propagating the error from the end to the beginning and during this back propagation I am considering the gradient the gradient operation we are considering so again corresponding to this neuron and we can determine the connecting weights $W_{(X1)3}$ and $W_{(X2)3}$ we can determine so I am repeating this during the back propagation I am moving from the output side to the input side that means I am moving in this direction and I am computing the gradients so I am computing the gradients and based on this the width updation rules I am updating the weights so that is the concept of the back propagation training algorithm and after this again we are determining this weights $W_{(X1)4}$,$W_{(X2)4}$,$W_{(X3)4}$ so all the weights I can determine and again $W_{(X1)5}$,$W_{(X2)5}$,$W_{(X3)5}$ so all the weights I am determining and finally I am determining the weights between the output node and the hidden layer 2 so the weights between the hidden layer 2 and the output node that is also determined that is nothing but W'46 and W'56 so this we have determined so this is the concept of the back propagation training algorithm so let us consider one numerical problem here the numerical on back propagation so suppose in

this network I have shown one network that is the feed forward network and I have two inputs X1 and X2 and we are considering the bias input is 1, 1 is the bias input so the bias values are B1 and B1 this two bias we are considering B1 and B1 so it is a very simple network and another bias is also considered this is the bias input and corresponding to this bias the weight is B2 that is the bias weight is B2 and in this case two input nodes two nodes in the hidden layer that is H1 and H2 and one node in the output layer that is Y1 so corresponding to this input X1 = 0.05 and X2 = 0.1 the target output is Y that is equal to 0.01 and we are considering the sigmoid activation function so that means corresponding to the input X1 is equal to 0.05 and X2 is equal to 0.1 the target output corresponding to the node Y = 0.01 so that information is available. So first I have to initialize the weights of the artificial neural network and generally it is randomly initialized so initial weights we are considering corresponding to W1, W2, W3, W4, W5, W6 all the weights so suppose the values are 0.15, 0.20, 0.25, 0.30, 0.40, 0.45 and the bias B1 and B2 are suppose 0.35 and 0.60 so these are the weights and during the forward pass that is the forward pass first I have to do the forward pass and after this I have to do the backward pass to adjust the weights of the artificial neural network. So first I am computing in_{H1} so that is nothing but corresponding to this hidden layer and the hidden node we are determining the in_{H1} that is nothing but W1 × X1 + W2 × X2 + B1 × 1 the bias input is considered B1 ×1 so if I put all these value that will be 0.3775 and we are determining the output activation at H1 that is nothing but we have to consider the sigmoid function $\frac{1}{1+e^{-\text{inH1}}}$ and if I put all these value I will be getting 0.593 and similarly we can determine out_{H2} also we can determine corresponding to H2 also we can determine the output that is out_{H2} we can determine that is equal to 0.596. Now in_o let us consider what is in_o so considering the H1, H2 and the bias input we can determine in_o that is the input to the output layer so we are considering now this output node and corresponding to this output node I am determining in_o that is nothing but W5 × out_{H1} + W6 × out_{H2} + B2 × 1 so if I put this value I will be getting this one 1.105 and we can determine the out_o we can determine so output we can determine corresponding to this network. Now we can determine the error so error is nothing but the target value minus the actual output value so that is the error the target value minus actual output value and if I put all these value the error will be 0.274 so that error I have to reduce now so I have to minimize the error so that is corresponding to the forward pass. So now let us consider the backward pass so corresponding to the output layer I have to find how much change in the weight W5 affects the error E so that is why we are considering the gradient of E with respect to W5.

So we have to determine the gradient of E that is the E is the error function that is the cost function with respect to the weight W5 and by considering the chain rule I can get this expression. So this $\frac{\partial E}{\partial out_o}$ by considering this I can determine this differentiation

$\frac{\partial out_o}{\partial in_o}$ that also I can determine and $\frac{\partial in_o}{\partial W_5}$ that also I can determine so values are 0.741, 0.186 and 0.593 and finally I can determine the value the $\frac{\partial E}{\partial W_5}$ that is equal to 0.082 and similarly I can determine the weight W6 also I can determine so that is nothing but 0.408. So like this during the back propagation I can determine the weights W6, W5 all the weights I can determine and after this let us consider the hidden layer and the input layer during the back propagation. So now we are considering the weights W1, W2, W3, W4, W3, W4 so these are the weights between the input nodes and the hidden nodes. So now we have to update the weights W1 so that is why this $\frac{\partial E}{\partial W_1}$ that I am determining and with the help of the chain rule we can obtain this.

So $\frac{\partial E}{\partial out_{H1}}$ we are determining by this expression so we can determine and the value is 0.55 and similarly $\frac{\partial out_{H1}}{\partial in_{H1}}$ that also we can determine that is equal to 0.241, $\frac{\partial in_{H1}}{\partial W_1}$ that is 0.05 and from this I can obtain $\frac{\partial E}{\partial W_1}$ that is 0.0006 so that is 0.0006. So based on this I can determine the weights W1, W2, W3 and W4. So similarly we can find the other weights W2, W3, W4. So you can see during the back propagation I am obtaining the values of the weights that is I am adjusting the values of the weights W1, W2, W3, W4, W5 and W6. So this is the concept of the back propagation training.

So now let us discuss the concept of the vanishing gradient. So what is this problem? So the vanishing gradients in deep neural networks. So in the figure I have shown two cases one is the training another one is the testing and I have shown the iterations in both the figures. So in the first case I am considering a 20 layer deep neural network and in the second case I am considering a network having 56 layers. So corresponding to the first network that is the 20 layered neural network so you can see the training error.

So it goes on decreasing with number of iterations. So you can see the training error corresponding to the 56 layer deep neural network and also you can see the training error corresponding to 20 layered deep neural network. The training error is significant corresponding to 56 layered deep neural networks. Similarly in the case of the testing also the performance of the 56 layered neural network is not good as compared to 20 layered deep neural networks. So this is because of the problem of the vanishing gradient.

So what is this problem? So for explaining this problem let us move to the next slide. So the vanishing gradient problem. So to explain this concept I am considering a simple network having 4 layers. So a network is considered and this F1, F2, F3 these are the activations corresponding to these 4 layers. This is the first layer, this is the second

layer, third layer and this is the fourth layer. The input to the network is X and I have shown the connections here and output of the network is O, O is the output of the network and I have to consider the weights the connecting weights.

So W1, W2, W3, W4 so these are the weights corresponding to this 4 layered network. So we are assuming that this W1, W2, W3, W4 these are scalars and the activation functions are also scalar. So that means the corresponding output will be also scalar. So what will be the output of the network? The output of the network is O. So I can write in this form so F4(W4 F3(W3 F2(W2 F1(W1 X)))).

So the output of the network I can write like this. So this expression I will be getting like this what will be the output of this node or the output of this network corresponding to this point. So output of the network is activation is F1 and it is W1 X. So like this I can compute the outputs of the nodes. So outputs of different layers I can compute like this and O is the final output.

So for mathematical convenience I will be representing like this. So this W1 X that I can consider as θ_1 and this W2 up to this point this I can consider as θ_2. So this up to this point I can consider it as θ_3 and up to this point this I am considering as θ_4. This is for the mathematical convenience I am considering. So if you see this expression what is this expression? The output of the network that is actually F4(W4 F3(W3 F2(W2 F1(W1 X)))) that is the output of the network.

So what is actually these outputs? The output is O. So in terms of θ I can write like this O = F4 (θ_4). This θ_4 = W4 F3(θ_3). θ_3 = W3 F2 (θ_2). θ_2 = W2 F1 (θ_1) and finally the θ_1 = W1 X. So I will be getting these values or I will be getting these outputs corresponding to the θ_1, θ_2, θ_3, θ_4.

So O = F4 (θ_4). This θ_4 = W4 F3(θ_3). θ_3 = W3 F2 (θ_2). θ_2 = W2 F1 (θ_1) and finally the θ_1= W1 X. So I can get this output. So now let us move to the next slide. So again I am writing the previous equation the previous equation I am writing again.

So that is F4(W4 F3(W3 F2(W2 F1(W1 X)))).. So the previous equation was like this and we considered this as θ_1, this as θ_2 this as θ_3 and finally this as θ_4. So argument θ_4, argument θ_3, argument θ_2, argument θ_1 I can consider like this. This is for the mathematical convenience. So corresponding to a particular network already I have explained during the back propagation I have to adjust the weights of the artificial neural network.

So that is why the error is back propagated to the input that means the gradient is back propagated to the input. So if I want to update a particular weight suppose W1 then I have to determine the gradient of the output with respect to the weight W1. So the

gradient I have to compute because the gradient value is back propagated to the input to adjust the weights of the artificial neural network. So this is the principle of the back propagation training. So now corresponding to this case suppose if I want to update the weight W1 so for this I have to determine this gradient. So by using the chain rule this $\frac{\partial O}{\partial W_1}$ that I can write by using the chain rule $\frac{\partial O}{\partial \theta_4}\frac{\partial \theta_4}{\partial F_3}\frac{\partial F_3}{\partial \theta_3}\frac{\partial \theta_3}{\partial F_2}\frac{\partial F_2}{\partial \theta_2}\frac{\partial \theta_2}{\partial F_1}\frac{\partial F_1}{\partial \theta_1}\frac{\partial \theta_1}{\partial W_1}$.

So which can be written like this X. F1'. W2. F2'. W3. F3'. W4. $\frac{\partial O}{\partial \theta_4}$. So I can write like this. So this actually you can write like this if you see this one this is nothing but X if you see this one this is nothing but F1'. So like this I am getting these values so all these values I am getting like this.

This is for the weight W1 so I have to determine the gradient of O with respect to W1. Similarly if I want to adjust the weight W2 I have to determine or I have to estimate this gradient of O with respect to W2. So that is nothing but $\frac{\partial O}{\partial \theta_4}\frac{\partial \theta_4}{\partial F_3}$. So this is by using the chain rule $\frac{\partial F_3}{\partial \theta_3}\frac{\partial \theta_3}{\partial F_2}\frac{\partial F_2}{\partial \theta_2}\frac{\partial \theta_2}{\partial W_2}$. So corresponding to this I can write in this from F1. F2'. W3. F3'. W4. $\frac{\partial O}{\partial \theta_4}$.

So I can write in this form. So in this case if you remember in case of the artificial neural network we considered the sigmoid activation function. So suppose the sigmoid activation function is F(x) and if I take the differentiation of the sigmoid activation function that is nothing but F'(x) = F(x)(1 - F(x)). So this is the derivative of the sigmoid activation function and it is maximum so it is maximum at x = 0 and corresponding to this the maximum value will be 1/4.

So the maximum value is 1/4 corresponding to this sigmoid function. So it is less than 1. So if you see this expression so we are multiplying this all the activation functions that is the derivative of the activation function that is the F1'. F2'. So these are mainly the derivative of the sigmoid activation function in case of the artificial neural networks or in case of the deep networks we randomly initialize weights during the back propagation. During the training of the artificial neural network or the deep networks we considered back propagation training and for this we randomly initialize weights of the network. So we can assume that these weights can come from a distribution function with the mean 0 and the variance 1. So I am repeating this so weights are randomly initialized and we can assume that these weights come from a probability distribution function with mean 0 and the variance is equal to 1.

So if you see this equation here you can see this is nothing but the product of F1'. F2'. F3' and already I told you the weights are from a probability distribution function with mean 0 and the variance 1. So because of this this product will be less than 1 and it will

go on decreasing if I increase the number of layers. So as we move towards the earlier layers for weight updation the number of products go on increasing. So if I compare this equation equation number 1 and equation number 2 suppose so number of products in equation 1 is more than the number of products in equation number 2 because we are considering the earlier layers.

So I am repeating this as we move towards the earlier layers for weight updation the number of products go on increasing. So because of this since we are doing the multiplications of the sigmoid activation function that is the derivative of the sigmoid activation function the gradient will go on decreasing decreasing with each and every layers. So if I compare the equation number 1 and equation number 2 so the gradient value will be less in case of the equation number 1 because we are considering the earlier layer that is W1 we are considering that is the first layer we are considering but in the equation number 2 we are considering the second layer. So as we move towards earlier layers the number of products go on increasing.

So that is the concept of the vanishing gradient. So the gradient value will go on decreasing. So what is the weight updation rule? So what is the weight updation rule if you remember that weight updation rule. So we have considered this we have considered this weight updation rule so $W(t+1)$ that is the weight at the $(t + 1)$ iteration and this is the weight in the t th iteration minus the learning rate η and we are computing the gradient. So we are computing the gradient. So since this gradient value will go on decreasing so this will go on decreasing then what will happen the weight at $(t + 1)$ iteration will be same as that of the weight at the t th iteration.

So I am repeating this since the gradient of W it will go on decreasing with increasing number of layers. So that is why the weight at $(t + 1)$ iteration will be not updated because it will be same as that of the weight at the t th iteration. So this will be negligible this term will be negligible because the gradient of the W that will be very very small. So that is why the weight will not be updated the weight at the $(t + 1)$ iteration will be same as that of the weight at the t th iteration. So weight at $(t + 1)$ iteration will be same as that of the weight at t th iteration that means the weights will not be updated.

Because the gradient value will go on decreasing with increase number of layers. So if I increase the number of layers the gradient value will go on decreasing because of this problem and opposite is the exploding gradient. So if the gradient value is very high so what will happen it will go on increasing and because of this it is very difficult to get the convergence condition and oscillations will take place. So that is the concept of the exploding gradient that is the opposite of the vanishing gradient. And the vanishing gradient because of the number of layers if I increase the number of layers the gradient value will go on decreasing and because of this the weight will not be updated the

weights will not be updated. But in case of the exploding gradient the gradient value is greater than 1 and because of this the gradient value will go on increasing and because of this it is difficult to get the convergence condition and the oscillation will take place.

So how to consider this issue and this vanishing gradient problem so how to consider this issue. So let us move to the next slide. So to consider this issue how to consider the first point is the choice of activation function ReLU instead of sigmoid function instead of the sigmoidal function. So in case of the deep networks we are considering more number of layers as compared to artificial neural networks because we are considering the deep architecture. So to consider this problem the problem of the vanishing gradient we may consider ReLU activation function instead of the sigmoidal activation function.

So that is one option. The second also what we can consider appropriate initialisation of weights. So suppose I have a network so n number of terminals are there. So now how to consider the appropriate weights. So in this case we can consider the distribution 0 mean distribution we can consider but variance we can consider as 1/n we can consider. So in the earlier case we considered the variance is 1 corresponding to this distribution because the randomly we are initialising the weights and corresponding to this we can assume that these weights are coming from a probability distribution function with mean 0 and variance 1. But in this case what we are considering the distribution having the mean 0 and the variance is equal to 1/n and sometimes also 2/n is also considered.

So either 1/n or 2/n we can consider for this initialisation of the weights. So variance will be 1/n or the 2/n and the mean will be 0. And finally another technique is that I am not going to discuss that is the intelligent back propagation learning algorithm. So we may employ this technique that is the intelligent back propagation learning algorithm we can also consider. So with the help of these techniques and that is the one important technique is the ReLU activation function.

ReLU activation function we can consider instead of the sigmoidal activation function. That is why the ReLU function is popular in case of the deep neural networks instead of the sigmoidal activation function. In this class I explained the concept of the back propagation training and also I explained the problem of the vanishing gradient and the exploding gradient. So if I increase the number of layers of a particular network the performance of the network may not improve because of the problem of the vanishing gradient and the exploding gradient. To consider this issue we may consider the ReLU activation function instead of the sigmoidal activation function. Also we can appropriately select the weights of the network that already I have explained and also we can consider the intelligent back propagation algorithm. So these are the techniques to

consider the vanishing gradient problem or the exploding gradient problem. So let me stop here today. Thank you.