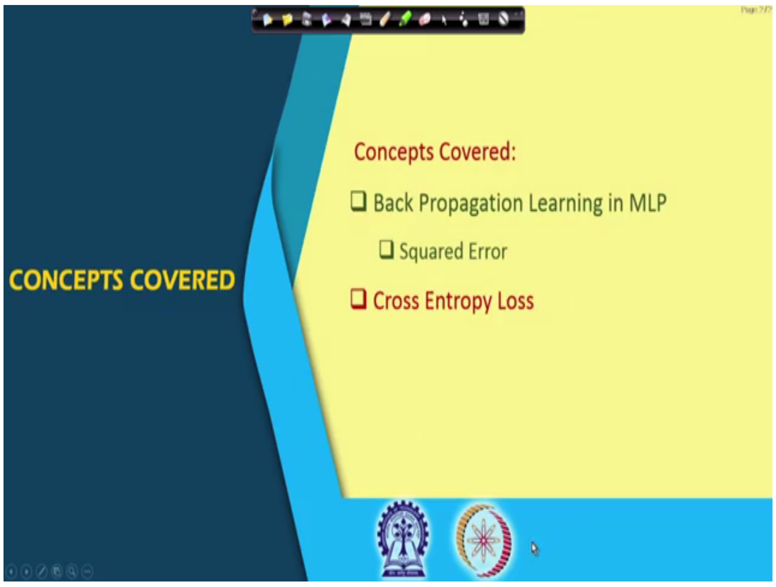
Deep Learning

Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 24
Loss Function

Hello, welcome to the NPTEL online certification course on Deep Learning. In our previous lecture, we were discussing about the back propagation learning in feed forward neural network or multilayer neural network and you remember that the loss function or the error function to be minimized that was considered in our previous lecture was sum of squared error or which is also known as quadratic error.

(Refer Slide Time: 01:00)

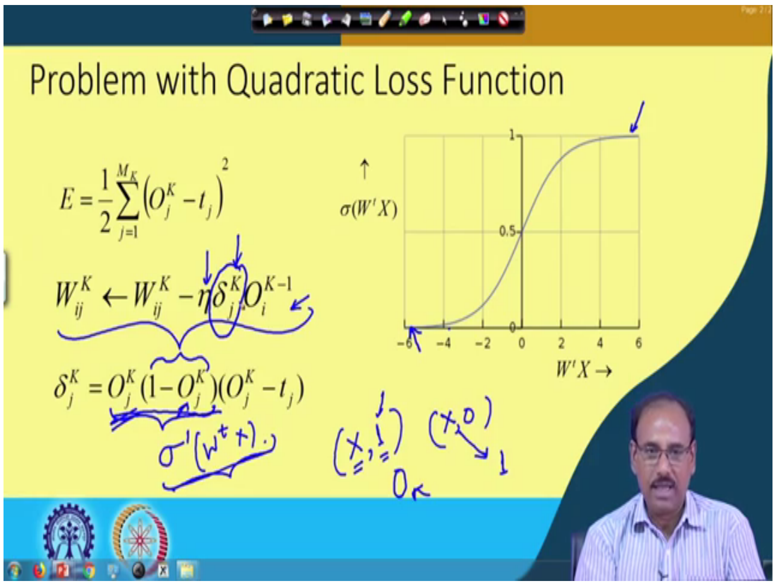


CONCEPTS COVERED

Concepts Covered:

- ❑ Back Propagation Learning in MLP
  - ❑ Squared Error
- ❑ Cross Entropy Loss

(Refer Slide Time: 01:02)



Problem with Quadratic Loss Function

$$E = \frac{1}{2}\sum_{j=1}^{M_K}\left(O_j^K - t_j\right)^2$$

$$W_{ij}^K \leftarrow W_{ij}^K - \eta\delta_j^K O_i^{K-1}$$

$$\delta_j^K = \underbrace{O_j^K(1-O_j^K)}_{\sigma'(W^t X)}(O_j^K - t_j)$$

So, the error function which was considered was given by E is equal to half of O j K minus t j square, where the summation is taken over j is equal to 1 to M K that is all the nodes in the output layer. Now, we will see that what are the problems with this quadratic loss function or squared error function, and we will try to find out or we will try to investigate that what alternative loss function that can be used to avoid the problem which is given by this quadratic loss function.

So, for updation of the weight rules, you find that the updation rule that we have used is W i j K gets W i j K minus eta times delta j K times O I K minus 1 in case of quadratic loss function. Where this delta j K was derived from the derivative of the sigmoidal output right, so we got delta j K as O j k, O j K is obviously, the output of the jth node in the kth layer, and we are considering the layer to be the output layer here. So, it is O j K into 1 minus O j K into O j K minus t j.

So, in appreciate that when this error updation rule or weight updation rule is necessary. If I feed a training vector, and I get the output if I find that the vector is correctly classified, obviously, I need not go for weight updation. So, I have to weight update the weights only when I find that I have fed an input vector which may belong to say class i, but my classifier is misclassifying that to class j.

So, only in such cases when the decision given by the classifier is wrong, I need to update the weight vector. As long as the decision is correct, the weight vectors need not be updated. And for updation of the weight vectors, when the moment you get an error is obtained by gradient descent rule as is given over here. So, as is given in this particular case.

And the one that is involved in the weight updation rule that is derived from the gradient is delta j K. So, you find that this O j K into 1 minus O j K is nothing, but derivative of the sigmoidal function which is W transpose X. Now, let us consider a case that I have an weight vector X, which actually belongs to class say 1. So, this was the training pair which was given that weight vector X, it actually belongs to class 1. So, when it is classified by this classifier the output of the node, if I consider I have a single neuron at the output say it is a two class problem

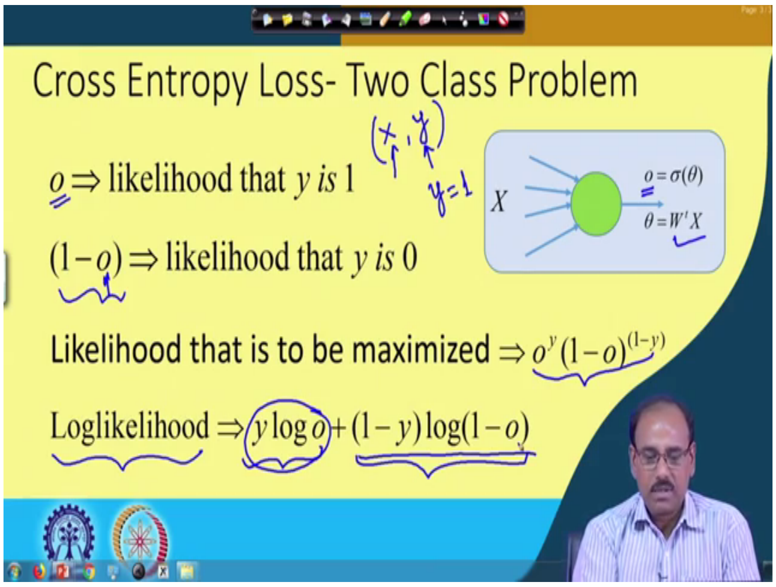
So, output of the neuron should be 1 or close to 1. If the output of the neuron is not 1 or say it is very close to 0, that means, I have an error right and because I have an error I have to go for updation of the weight vectors by propagating this error in the backward direction, and that is where this gradient of the output that comes into picture.

So, here you find that actually my y should be equal to 1 but I am getting and y which is equal to 0 or near to 0 and that comes over here, my output is O j K times 1 minus O j k. So, this product O j K minus 1 into 1 minus O j K becomes very very low. Similarly, in the other case, if a training vector is given as belonging to 0, but the classified classifies that two class one, that means, my output of the neuron should actually be 0.But the classifier has given a very high output that means, it is the miss classification. Again in this case I have to go for updation of the weights following the same weight updation rule. And here you find that O j K as decided by the classifier being very high 1 minus O j K will be very low.

And that is because if you look over here when your output is very low in my sigmoidal function, I am somewhere over here; when the output is very high in the sigmoidal function, I am somewhere over here and in both these regions, your derivative of the sigmoidal function that is sigma dash W transpose X that is very very low and in the extreme case, it may even vanish. So, the gradient vanishes. And if the gradient vanishes or the gradient is very very low, you find that this gradient is directly influencing the rate of training, because your rate of training is controlled by not only the convergence rate eta, it is also controlled by delta j K.

So, if O j K minus O j K into 1 minus O j K whether O j K is 0 or O j K should be 1, whatever the case may be, if any of the terms is very low, then your rate of learning becomes very very low. So, that is the effect of or the bad effect of this quadratic loss function that we have. So, is there any remedy of this? So, let us try to see that whether any other loss function can avoid this tendency of slow training or slow learning.

(Refer Slide Time: 07:54)



Cross Entropy Loss- Two Class Problem

(x, y) $y=1$

$o \Rightarrow$ likelihood that y is 1

$(1-o) \Rightarrow$ likelihood that y is 0

Likelihood that is to be maximized $\Rightarrow o^{y}(1-o)^{(1-y)}$

Loglikelihood $\Rightarrow y \log o + (1-y)\log(1-o)$

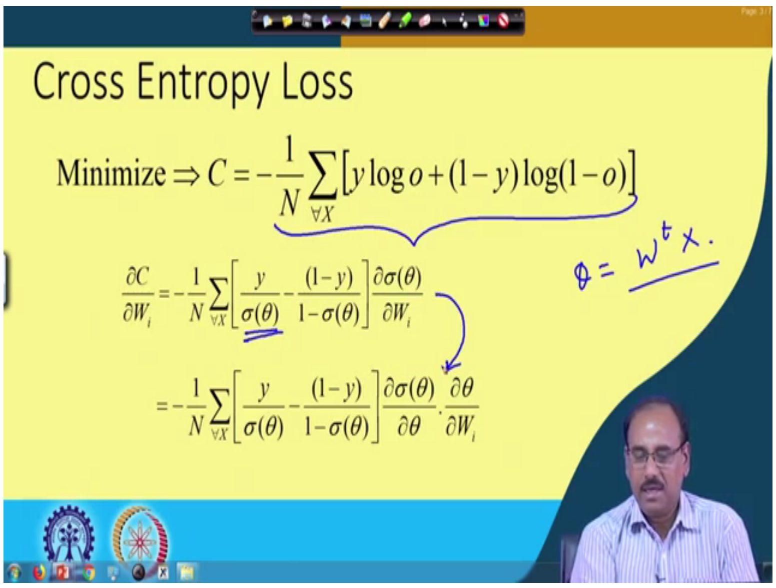
So, here comes another loss function which is called cross entropy loss. So, how do you define this cross entropy loss? Again I am taking a two class problem So, if a feature vector X, so again I am assuming that my input training vectors are given by given as ordered pairs x, y where x is the input vector and y is the ground truth that is the actual class to which this vector X belongs this is given for training purpose right.

So, if y is actually equal to 1m that means, I get our training vector from class one for which output should be equal to 1. Whereas output of your neural network is o, so whatever is the output, this output actually gives you the likelihood that y is 1. In the same way if y is equal to 0, that means, the training vector belongs to class belongs to another class, then 1 minus o, where o is the output of the neuron that gives you the likelihood that y is 0. So, I can combine these two to get a likelihood that needs to be maximized which is given by o to the power y into 1 minus o into 1 minus o to the power 1 minus y. So, this is the likelihood that needs to be maximized for training this neural network.

And from here of course, you find that an exponent is involved in this expression, and I do not like exponents. So, how do we, how do we avoid the exponents? So, obviously, you take instead of the likelihood you take the log likelihood. So, the log likelihood simply becomes y log o, where o is the output of the neuron ok. Again here I am assuming that the output is a sigmoidal function of the weighted sum of the inputs as given over here. So, I get this log likelihood which is given by y log o into 1 minus y log o minus y log 1 minus o.

So, here you find that if y is equal to 1, in that case this term that is if the training sample X is taken from class omega 1, then I should get y equal to 1 right in that case 1 minus y is 0. So, what I have to maximize is y log o. In the other case, if y is equal to 0, that means, the training sample has been taken from other class. So, this term y log o becomes 0. So, what I have to maximize is 1 minus y log 1 minus o. So, this is what is the likelihood that I need to maximize.

(Refer Slide Time: 11:03)



Cross Entropy Loss

$$\text{Minimize} \Rightarrow C = -\underbrace{\frac{1}{N}\sum_{\forall X}\left[y\log o + (1-y)\log(1-o)\right]}$$

$$\theta = W^t X.$$

$$\frac{\partial C}{\partial W_i} = -\frac{1}{N}\sum_{\forall X}\left[\frac{y}{\underline{\sigma(\theta)}} - \frac{(1-y)}{1-\sigma(\theta)}\right]\frac{\partial \sigma(\theta)}{\partial W_i}$$

$$= -\frac{1}{N}\sum_{\forall X}\left[\frac{y}{\sigma(\theta)} - \frac{(1-y)}{1-\sigma(\theta)}\right]\frac{\partial \sigma(\theta)}{\partial \theta}\cdot\frac{\partial \theta}{\partial W_i}$$

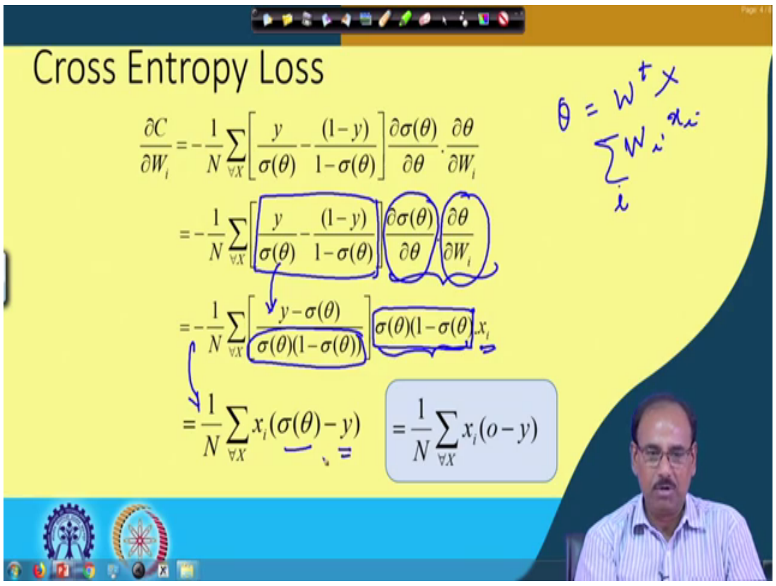
So, once I have this log likelihood, from here I can find out the cross function I can define a cross function which is minus y log o into 1 minus y log 1 minus o, take the sum over this, sum of this over all the input feature vectors and then you take the average. So, I am defining a cost function C to be minus of 1 by N summation of y log o plus 1 minus y log 1 minus o, taking the summation of this over all input vectors, and taking the average which is given by 1 by N and this is what is known as cross entropy loss.

So, you find that there in the previous case, when we talked about we said that we want to maximize this log likelihood which is y log o plus 1 minus y log 1 minus o, which is equivalent to minimization of the function C which is our cross entropy loss. So, again as before for minimization of this cross entropy loss we have to follow the gradient descent approach. So, I need to take the derivative of this cross entropy loss C with respect to the weight vector or the gradient of C with respect to weight vector W, and I can do it partially. So, you take the derivative of this cross entropy loss C with respect to W i that is the ith component of the weight vector W.

So, what I compute is del C del W i, and from here you find that when I take the derivative of this cross entropy loss C, the derivative of C with respect to W i, what I get is y by sigma theta as we have said earlier there is this theta is nothing but W transpose X. So, I get 1 upon sigma theta minus 1 minus i 1 minus y upon 1 minus sigma theta into del sigma theta del W i because this is a logarithmic function. So, it is actually 1 upon o, and o is nothing but sigma theta.

So, del C del W i simply becomes 1 upon n sum of y upon sigma theta minus 1 minus y upon 1 minus sigma theta into del sigma theta del W i take the summation of this over all input vectors X and that simply becomes minus 1 upon N then sum of y upon sigma theta minus 1 upon y upon 1 minus sigma theta into again here I apply the chain rule. So, it is del sigma theta del theta into del theta del W i simply using the chain rule.

(Refer Slide Time: 14:20)



Cross Entropy Loss

$$\frac{\partial C}{\partial W_i} = -\frac{1}{N}\sum_{\forall X}\left[\frac{y}{\sigma(\theta)} - \frac{(1-y)}{1-\sigma(\theta)}\right]\frac{\partial \sigma(\theta)}{\partial \theta}\cdot\frac{\partial \theta}{\partial W_i}$$

$$= -\frac{1}{N}\sum_{\forall X}\left[\boxed{\frac{y}{\sigma(\theta)} - \frac{(1-y)}{1-\sigma(\theta)}}\right]\frac{\partial \sigma(\theta)}{\partial \theta}\cdot\frac{\partial \theta}{\partial W_i}$$

$$= -\frac{1}{N}\sum_{\forall X}\left[\frac{y-\sigma(\theta)}{\sigma(\theta)(1-\sigma(\theta))}\right]\boxed{\sigma(\theta)(1-\sigma(\theta))}\,x_i$$

$$= \frac{1}{N}\sum_{\forall X} x_i(\sigma(\theta)-y)$$

$$\boxed{= \frac{1}{N}\sum_{\forall X} x_i(o-y)}$$

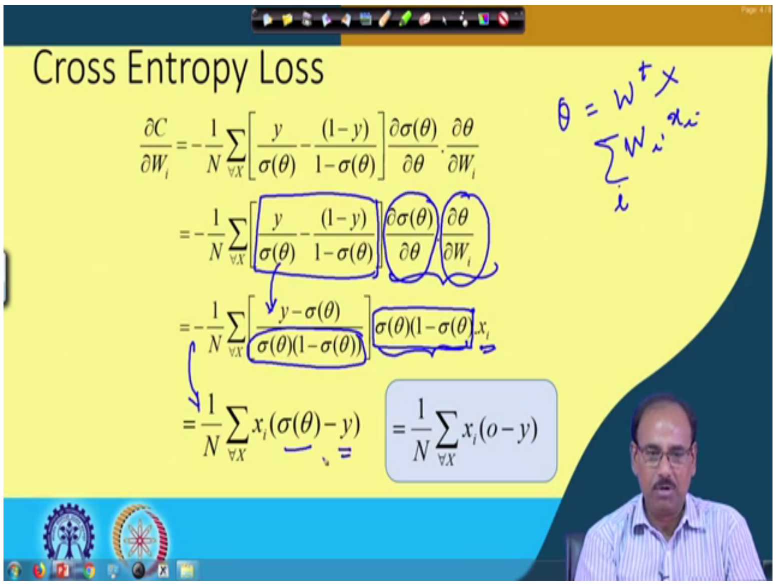
$$\theta = W^t X$$
$$\sum_i W_i x_i$$

So, if I go for simplification further, the same expression is written as minus 1 upon N, then sum of 1 upon sigma theta minus 1 minus y upon 1 minus sigma theta into del sigma theta del theta into del theta del W i which simply becomes if I go for simplification of this, it simply becomes minus. So, here you find that del sigma theta del theta, there is del sigma theta del theta as it is a sigmoidal function is nothing but sigma theta into 1 minus sigma theta.

And del theta del W i as you remember that del theta was W transpose X. So, del theta del W i simply becomes x i, because it is nothing but W i times x I take the summation over all i. So, it simply becomes X i. So, this term del sigma theta del theta into del theta del W i is simply sigma theta into 1 minus sigma theta into x i and when I simplify this term that is y upon sigma theta minus 1 minus y upon 1 minus sigma theta, the say expression simplified expression simply becomes y minus sigma theta upon sigma theta into 1 minus sigma theta.

So, from here you find that this sigma theta into and this sigma theta into 1 minus sigma theta, and this sigma theta into 1 minus sigma theta gets cancelled. So, my expression simply becomes 1 by N into x i times sigma theta minus y take the summation of this and then you take the average. So, my gradient del C del W i simply becomes 1 over N, then x i o minus y, where o i is nothing but sigma theta take the summation over all input vector X that is what is my del C del W i.

(Refer Slide Time: 16:56)



Cross Entropy Loss

$$\frac{\partial C}{\partial W_i} = -\frac{1}{N}\sum_{\forall X}\left[\frac{y}{\sigma(\theta)} - \frac{(1-y)}{1-\sigma(\theta)}\right]\frac{\partial \sigma(\theta)}{\partial \theta}\cdot\frac{\partial \theta}{\partial W_i}$$

$$= -\frac{1}{N}\sum_{\forall X}\left[\boxed{\frac{y}{\sigma(\theta)} - \frac{(1-y)}{1-\sigma(\theta)}}\right]\frac{\partial \sigma(\theta)}{\partial \theta}\frac{\partial \theta}{\partial W_i}$$

$$= -\frac{1}{N}\sum_{\forall X}\left[\frac{y-\sigma(\theta)}{\sigma(\theta)(1-\sigma(\theta))}\right]\boxed{\sigma(\theta)(1-\sigma(\theta)}.x_i$$

$$= \frac{1}{N}\sum_{\forall X} x_i(\sigma(\theta)-y)$$

$$\boxed{= \frac{1}{N}\sum_{\forall X} x_i(o-y)}$$

$\theta = W^t X$

$\sum_i W_i x_i$

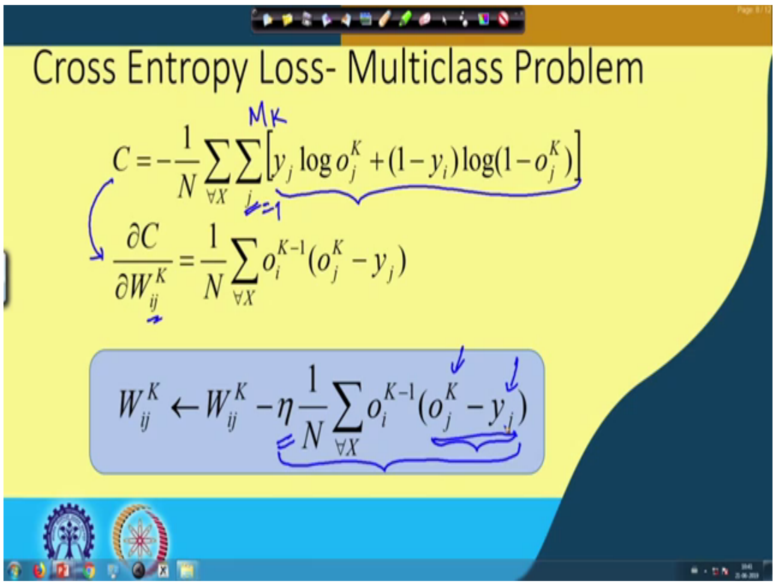
And once I have this del C del W i, I can just write the weight updation rule which is nothing but W i gets W i minus eta times 1 by N sum of x i into o minus y, where the summation has to be taken over all the feature vectors input feature vectors. Now, what is the advantage that we get? You find that in case of cross entropy in case of the squared error loss or the quadratic loss. In the updation term, we had a term sigma theta into 1 minus sigma theta which is nothing but derivative of sigma theta with respect to theta.

And this sigma theta into 1 minus sigma theta these terms were responsible for slow learning, because if sigma theta is very high very high means it is very near to 1, 1 minus sigma theta almost vanishes. On the other hand, if I am on the other side that is sigma theta is almost equal to 0, then also the derivative term almost vanishes. And because of the vanishing of the derivative, the learning becomes very very slow.

Whereas in this case in my updation term I simply have o minus y i times x i, where o is the output and y i is the ground truth. So, here you find that if my ground truth is actually 0, that means, if y is equal to 0, whereas, if I get o to be very high say 0.99 something like this, then o minus y is high unlike in the previous case. Similarly, if my y is 1, and o is the 0.001, it is very low. Then again o minus y, the absolute value is very high, that means, my rate of learning is now proportional to the difference of the output that you get and what is my ground truth.

So, if the error is more, the rate of learning is more, which is in contrast to what we have obtained in case of quadratic error that if error is very, very high, your rate of learning becomes slow which is not desirable. So, here if your error is high, the rate of learning is also high, so that is the advantage of this cross entropy loss over the quadratic error loss. So, this is what we have got in case of a binary classifier or a two class problem and this cross entropy that we have defined as given by this term this is what is known as binary cross entropy that is y log o into 1 minus y log 1 minus o, this is what is known as binary cross entropy right.

(Refer Slide Time: 20:24)



Cross Entropy Loss- Multiclass Problem

$$C = -\frac{1}{N}\sum_{\forall X}\sum_{j=1}^{M_K}\left[y_j \log o_j^K + (1-y_j)\log(1-o_j^K)\right]$$

$$\frac{\partial C}{\partial W_{ij}^K} = \frac{1}{N}\sum_{\forall X} o_i^{K-1}(o_j^K - y_j)$$

$$\boxed{W_{ij}^K \leftarrow W_{ij}^K - \eta\frac{1}{N}\sum_{\forall X} o_i^{K-1}(o_j^K - y_j)}$$

So, what we have done is in case of a two class problem or binary classifier. How do we extend this to a multiclass problem? So, you remember that in a multiclass problem, I have at the output layer says C number of nodes where C is the number of categories or the number of classes and when I wanted to use this concept of entropy loss or cross entropy loss, then the outputs are to be defined or are to be obtained in a probabilistic sense. In the manner that o j should give me that what is the likelihood that y j is equal to 1 and for that we had defined something called soft max classifier, where soft max classifier gives you the normalized probability at the output that two what is the probability that an input vector x belongs to class omega j. So, that is what is given by soft max classifier.

So, when I want to use this cross entropy loss for training the feedback feed forward neural network at the output layer I assume that the output is a soft max output that means, the outputs are gives you the probability or belongingness of a of an input vector to a particular class. So, given this the same thing that we have defined before that we discussed before that if my y j is equal to 1, then O j K that is the output of the jth node at the output layer that is kth layer that gives you what is the likelihood of y j belonging being 1.

Similarly, 1 minus O j K tells you that what is the likelihood that y j K y j is 0. So, accordingly in the same definition of the binary cross entropy for every individual output node, I can use. So, for the jth node, the corresponding cross entropy is defined as y j log O j K plus 1 minus y j log 1 minus O j K take the negative of this, so that becomes the cross entropy or binary cross entropy corresponding to the jth layer jth node in the output layer.

And to get the overall cross entropy, what I have to do is, I have to take the sum of all these entropies all these cross entropies over all the nodes in the output layer, so that is the reason that. I take the summation over of this over all j, where j varies from 1 to M K, where M K is the nodes number of nodes in the output layer and this have to compute over all the feature vectors X.

So, I have this outer summation, you take the summation over all input feature vectors X. And again here as before if I take the derivative of C with respect to W i j K. Now, you find that I have multiple number of layers, I have multiple number of nodes in every layer. So, my index becomes i j that is from ith node in K minus first layer to jth node in the kth layer

So, I compute del C del W i j K and as before you can verify that the output will be 1 upon N summation of O I K minus 1, where O I K minus 1 is the output of the ith node in the K minus first layer. So, it is sum of O I K minus 1 into O j K minus y j, and you take the summation over all j and take the average.

So, accordingly your weight updation rule using this cross entropy loss function becomes W i j K getting W i j K minus eta times, again eta is your rate of convergence 1 over N summation of O I K minus 1 into O j K minus y j. And again as before you can find that more the difference between O j K and y j, O j K is your actual output that you are getting from the jth neuron in the kth layer, and y j K is the expected output or this is the ground truth.

So, if O j K minus y j is more that indicates that error is more and your updation component the value by which you want to update the weight vector is now directly proportional to O j K minus y j, that means, if the error is more your rate of learning is more; if the error is less that rate of learning is less. So, this is the advantage of using the cross entropy loss for training that the neural network over the quadratic error which is use for training the neural network but you have to remember that if I want to use this cross entropy loss, then output of the neural network must be a soft max output because it has to be a probabilistic measure.

We will stop here today. We will continue with other neural networks in future classes.

Thank you.