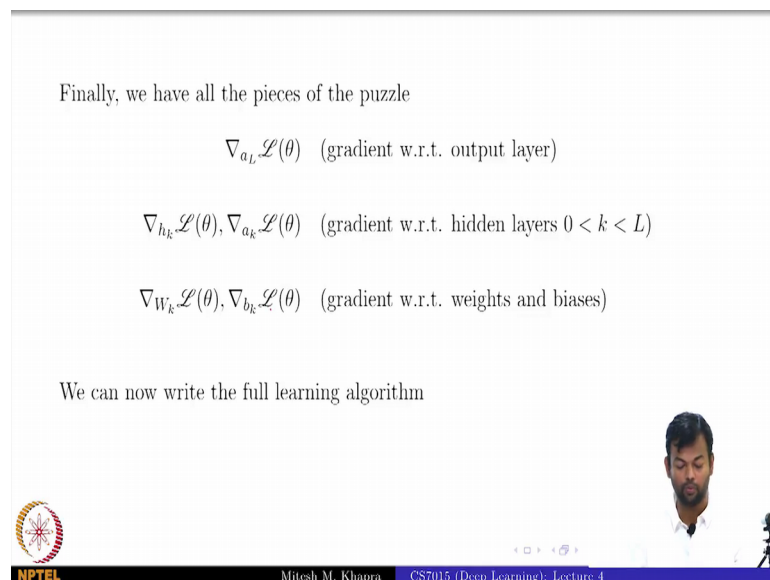


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 4.8
Lecture - 04
Back Propagation: Pseudo Code

So, we move on to the next module and now we will write Pseudo Code to for back propagation. And pay attention this is what your assignment is about. So, this is how you will write your assignment, ok.

(Refer Slide Time: 00:23)



Finally, we have all the pieces of the puzzle

$$\nabla_{a_L} \mathcal{L}(\theta) \quad (\text{gradient w.r.t. output layer})$$
$$\nabla_{h_k} \mathcal{L}(\theta), \nabla_{a_k} \mathcal{L}(\theta) \quad (\text{gradient w.r.t. hidden layers } 0 < k < L)$$
$$\nabla_{W_k} \mathcal{L}(\theta), \nabla_{b_k} \mathcal{L}(\theta) \quad (\text{gradient w.r.t. weights and biases})$$

We can now write the full learning algorithm

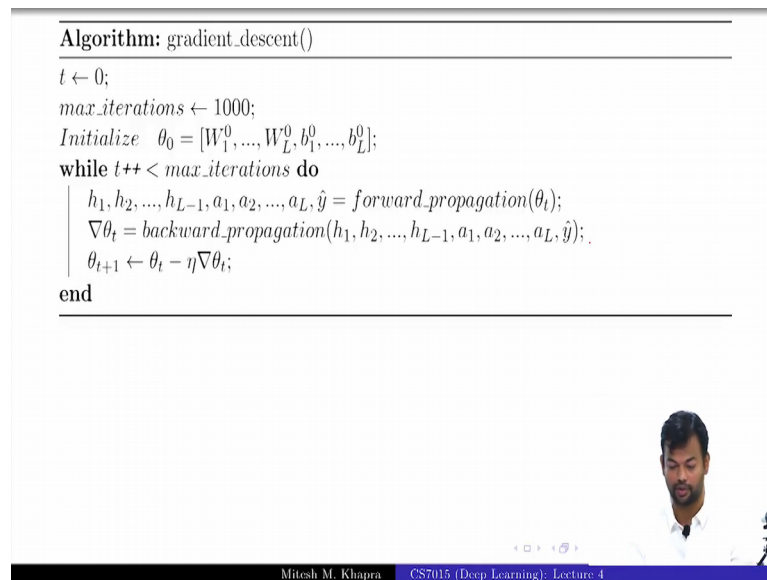
The slide includes a small video inset of the presenter in the bottom right corner, a navigation bar at the bottom with the NPTEL logo, the name 'Mitesh M. Khapra', and the course title 'CS7015 (Deep Learning): Lecture 4'.

So, we have all the pieces of the puzzle, we have the gradients with respect to the output layer, that was the spatial layer because, the output activation function is different. They are the gradients with respect to all the hidden layers; that means, I have the gradients with respect to the activations as well as the pre activation.

So, in the h's as well as the a's and I also have the gradients with respect to the weights and the biases and this is all index agnostic right; that means, I am just using k as the index everywhere, I have a generic formula, which applies at any layer for the weights as well as the activations and the pre activations right ok. Now, we can put all this together into a full learning algorithm. So, let us see what the pseudo code looks like.

(Refer Slide Time: 01:03)

```
Algorithm: gradient_descent()
t ← 0;
max_iterations ← 1000;
Initialize  $\theta_0 = [W_1^0, \dots, W_L^0, b_1^0, \dots, b_L^0]$ ;
while t++ < max_iterations do
     $h_1, h_2, \dots, h_{L-1}, a_1, a_2, \dots, a_L, \hat{y} = \text{forward\_propagation}(\theta_t)$ ;
     $\nabla\theta_t = \text{backward\_propagation}(h_1, h_2, \dots, h_{L-1}, a_1, a_2, \dots, a_L, \hat{y})$ ;
     $\theta_{t+1} \leftarrow \theta_t - \eta \nabla\theta_t$ ;
end
```



So, we have this t equal to 0 well run this for some max iterations we initialize all the parameters to some quantity will randomly initialize them ok. Now, for these max iterations, can you tell me what is the first thing that I will do? So, there will be 2 functions here ok. Tell me what those 2 functions would be.

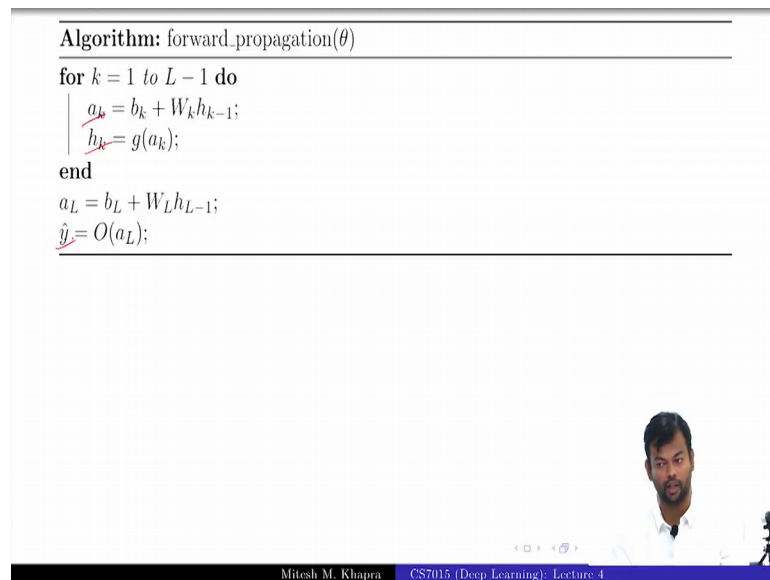
Student: Forward.

Forward propagation and then backward propagation right. So, you do a forward propagation and you compute all these activations pre activations output layer loss everything right and then you do this backward propagation where you feed all these things which you have computed right. These are the quantities which you have computed; you will pass this to your backward propagation algorithm it would not look. So, nasty as this it will not take. So, many parameters you could write it smartly and then you will just do the parameter update right.

So, what will the back propagation give you actually all the gradients all the partial derivatives right and then once you have the partial derivatives, you know how to compute the update law. Is this clear? So, now, let us look at these 2 functions more carefully the forward propagation and the backward propagation, ok.

(Refer Slide Time: 02:09)

```
Algorithm: forward_propagation( $\theta$ )
for  $k = 1$  to  $L - 1$  do
     $a_k = b_k + W_k h_{k-1}$ ;
     $h_k = g(a_k)$ ;
end
 $a_L = b_L + W_L h_{L-1}$ ;
 $\hat{y} = O(a_L)$ ;
```



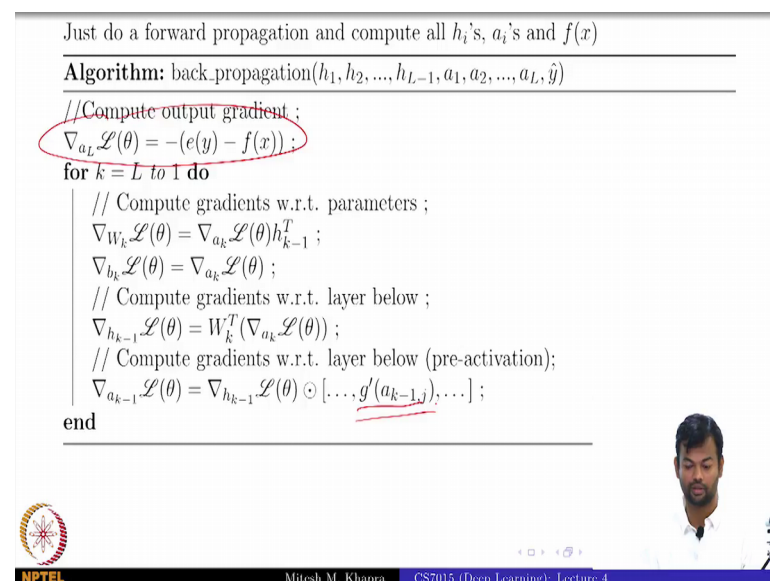
So, forward propagation is simple for all the hidden layers; that means, from layer 1 to layer $L - 1$ what will I do give me the code a k is equal to good then ok. And what is h of 0 you are starting the loop from 1 right. So, you will need h of 0 that is x and then you will have a special treatment for the output layer and your final output will be whatever output function you use ok. This makes sense you can write this in python ok.

You will have to write this in python, ok.

(Refer Slide Time: 02:45)

Just do a forward propagation and compute all h_i 's, a_i 's and $f(x)$

```
Algorithm: back_propagation( $h_1, h_2, \dots, h_{L-1}, a_1, a_2, \dots, a_L, \hat{y}$ )
// Compute output gradient ;
 $\nabla_{a_L} \mathcal{L}(\theta) = -(e(y) - f(x))$ ;
for  $k = L$  to 1 do
    // Compute gradients w.r.t. parameters ;
     $\nabla_{W_k} \mathcal{L}(\theta) = \nabla_{a_k} \mathcal{L}(\theta) h_{k-1}^T$  ;
     $\nabla_{h_k} \mathcal{L}(\theta) = \nabla_{a_k} \mathcal{L}(\theta)$  ;
    // Compute gradients w.r.t. layer below ;
     $\nabla_{h_{k-1}} \mathcal{L}(\theta) = W_k^T (\nabla_{a_k} \mathcal{L}(\theta))$  ;
    // Compute gradients w.r.t. layer below (pre-activation);
     $\nabla_{a_{k-1}} \mathcal{L}(\theta) = \nabla_{h_{k-1}} \mathcal{L}(\theta) \odot [\dots, g'(a_{k-1,j}), \dots]$  ;
end
```



Ah Now we have computed all the h 's and the a 's what have we computed all the a 's, all the h 's and all and the Y , right now you want to do back propagation. So, back propagation the loop will be from i equal to 1 to n minus 1 good. So, the first thing I will compute is the gradient with respect to the output layer. See, even here the output layer was outside the loop. The same thing would happen here also. In the back propagation also first you will compute the gradient with respect to the output layer and this is the formula.

If you remember from last class right, that is the formula which I have substitute here and note that f of x is known to you because you computed that in the forward pass and of y what is e of i y 1 hot vector which with a correct label said to 1 and you know what the correct label is because, we have given you the (Refer Time: 03:14) data right ok then what would the loop be 1 to 1 or 1 minus 1. Let us see first you compute the gradients with respect to parameters ok, ok. It is 1, correct.

So, because, we are using k minus 1, then you compute the gradients with respect to the layer below computes gradients with respect to the pre activation right. This is exactly how you will proceed this. Is clear to everyone? The same 3 components that we have used you might be a bit confused about the ordering in which we have put them right because we computed the gradients with respect to pre activation first and then the weights, but once you go back, you will realize because it is the way we have indexed it right because this is already outside.

So, this has already been computed. So, you can already compute the gradients with respect to the weights of the outermost layer, is that fine? So, this is straightforward you can go back and check this now anything remaining, or you have everything can you just take a minute and see if you can visualize the python code and we will just assume that you are done the assignment you can read you will have multiple these vectors and matrices and so on. And you are just doing a lot of matrix operations using (Refer Time: 04:06) or (Refer Time: 04:08) or whatever you prefer right, is that clear? Ok.

Now, what is missing here? Input is missing, ok. Input we have given right. The ominous data set has been given is there something that yours I have still not shown you how to compute; oh I did not update the parameters here is it? No the parameter update will happen in the outer loop, right. So, those forward prob back prob and then update the

parameters right. So, the main algorithm was forward prop back prob update the parameters, when we saw forward prob an obvious seeing backward prob, ok. So, what is missing 1000 iterations? Something in the last line before end of course, do you know how to compute this? You know how to compute this.