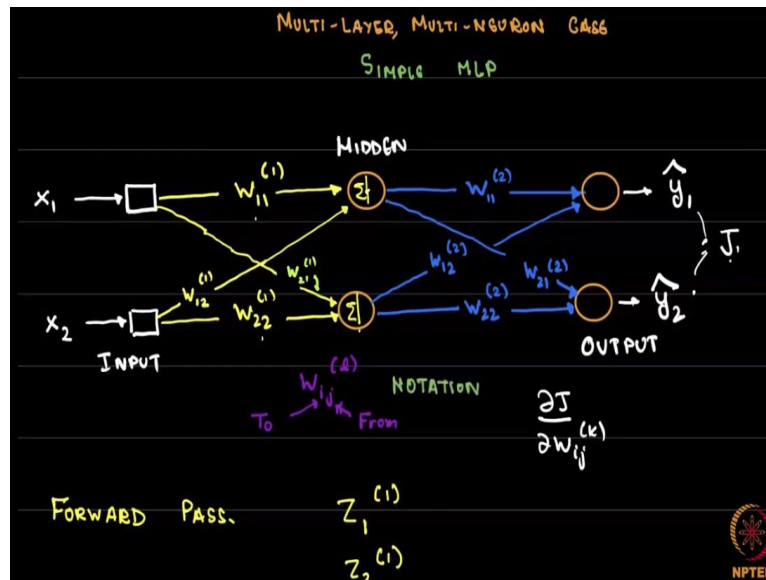


**Inverse Methods in Heat Transfer**  
**Prof: Balaji Srinivasan**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology – Madras**

**Lecture - 55**  
**Backprop in a MLP**

(Refer Slide Time: 00:19)

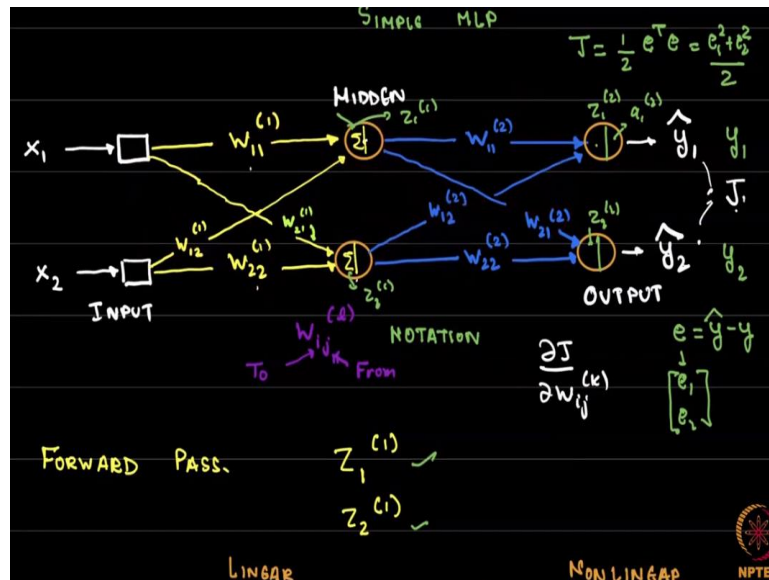


Welcome back. This is week 10 of Inverse Methods in Heat Transfer. We were looking at the back propagation algorithm in the last video and we will continue our exploration of the back prop for the multi-layer multi neuron case here. So, this is a simplified multi-layer perceptron as we will see this simplified case is actually something that is sufficient for us to write a full general algorithm for an MLP.

So, here it is what I have done here is as you can see, we have taken two input neurons, one hidden layer and two output neurons and this is sufficient unlike the previous case where we took two hidden neurons in a scalar chain you will see that once you understand this case fairly well it is fairly easy to understand what is exactly happening in the multi-layer perceptron case.

Now one thing about notation as I had mentioned earlier in the previous video also, we are starting with to and then from. So,  $i$  is the one where it is going to and  $j$  is where it starts for those of us who write from left to right this is kind of confusing.

(Refer Slide Time: 01:45)



So, for example, once again I have ignored all bias units as I did in the last video. we require this  $\frac{\partial J}{\partial w_{ij}^{(k)}}$ , but if it starts with the second neuron and goes to the first neuron here, I am calling it  $w_{12}$  and not  $w_{21}$  so  $w_1$  stands for this to here, this of course is  $w_{22}$  and this looks like  $w_{21}$ . So, we read from here rather than from there. So, all these superscript 1 simply mean.

This is the first layer superscript 2 means the second layer same thing  $w_{11}$  goes from the first to the first  $w_{21}$  goes from the first to the second  $w_{12}$  goes from the second to the first and  $w_{22}$  goes from the second to the second. Now you have two outputs here just to take a general case  $\hat{y}_1$  and  $\hat{y}_2$ . Remember now even though  $\hat{y}_1$  and  $\hat{y}_2$  are two different numbers  $J$  is still one.

You would have just like in the soft max case you have  $y_1$  and  $y_2$  let us say we are taking  $J$  as to the least square expression again we are going to taking  $\frac{1}{2} e^2$  or more precisely I will write it as  $e^T e$ , where  $e$  is the vector of  $(y - \hat{y})$  or  $(\hat{y} - y)$  let us keep it that way. Now notice  $e$  is going to have an  $e_1$  and it is going to have an  $e_2$ . So, when you do  $e$  transpose  $e$  this is the same as  $\frac{e_1^2 + e_2^2}{2}$ .

Now, all this is abstracted once we start using a matrix notation as you will see shortly. So, we want to do the forward pass. how would we do the forward pass? we already did a simple example somewhat similar to this in two videos ago. You take  $x_1$  and  $x_2$  pass it through this do a summation and you get  $z$  you get  $z_1$  here,  $z_2$  here,  $z_1$  will give  $a_1$ ,  $z_2$  will give  $a_2$ . Now you have  $a_1$  and  $a_2$  here multiply all these matrices.

You will get again a  $z_1$  and a  $z_2$  here maybe I should mark that here as well, but this one will be  $z_{12}$  and  $z_{22}$ . Similarly, this one will be  $z_1$  let us write it here set one and  $z_{21}$  and correspondingly we will have  $a_{12}$  etcetera. Now we want to calculate  $z_{11}$  and  $z_{21}$  that is the very first step of our forward prop. If you are given these two numbers  $x_1$   $x_2$  and guesses for all these weights that is the way you will go ahead.

(Refer Slide Time: 04:38)

The image shows handwritten notes on a blackboard background, divided into two sections: LINEAR and NONLINEAR.

**LINEAR:**

$$z_1 = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2$$

$$z_2 = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2$$

Below these equations, the word "MATRIX" is written with a downward arrow. A matrix equation is shown:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

The matrix of weights is labeled  $w^{(1)}$  and the input vector is labeled  $x = a^{(0)}$ . A boxed equation summarizes this as:

$$z^{(1)} = w^{(1)} \cdot x$$

The dimensions are noted as  $z^{(1)}$  being  $2 \times 1$ ,  $w^{(1)}$  being  $2 \times 2$ , and  $x$  being  $2 \times 1$ .

**NONLINEAR:**

$$a_1^{(1)} = g(z_1^{(1)})$$

$$a_2^{(1)} = g(z_2^{(1)})$$

A note "Elementwise nonlinearity" points to the function  $g$ . A diagram shows the mapping from the vector  $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$  to the vector  $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$  via the function  $a = g(z)$ .

The overall relationship is summarized as  $z^{(1)} = w^{(1)} a^{(0)}$ .

Now let us first do it in two steps just like we did the forward prop calculation. We are going to first calculate the linear. So, if I want  $z_1^{(1)} = w_{11}x_1 + w_{12}x_2$  that is what I have written  $w_{11}x_1 + w_{12}x_2$ . Similarly,  $z_2^{(1)} = w_{21}x_1 + w_{22}x_2$ . so that I have written here, but there is one neat thing that you can do at this point which was not so obvious.

When we did the scalar case, you can write this as a matrix. Now this comes out to be a nice matrix  $\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$ . So, we can basically write this as the matrix  $w^{(1)}$  this entire matrix  $w$  that I stand for level 1 and this of course is also a matrix multiplied by  $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$  that stands for  $x$  matrix or the  $a^{(0)}$  matrix. So, you can write  $z^{(1)} = w^{(1)}x$  or you can write  $z^{(1)} = w^{(1)}a^{(0)}$ .

Now, notice this is exactly the same as what we did in the scalar chain  $z_1 = w_1 a_0$  and we have the same expression here which is quite neat you have  $z^{(1)} = w^{(1)} a^{(0)}$ , this express exactly the same except now this is a 2 cross 1, this is a 2 cross 2 and this is a 2 cross 1 again. Next, we

are not just done with the linear part we have to calculate the a's, but the a's are easy.  $a_1^{(1)}$  is simply the non-linearity applied on  $w_1$  or  $z_1$ ;  $a_2^{(1)}$  is simply the non-linearity applied on  $z_2$ .

So, we simply say  $a = g(z)$  or we can say something like  $a_1 = g(z_1)$  where it is understood that if  $a$  is a matrix that  $g$  applies on each element. So, this is what is known as element wise operation. So, if  $g$  was a sin function so that would be sin of  $a_1$  is  $z_1$ ; sin of  $a_2$  is  $z_2$ , if  $g$  is sigmoid then sigmoid of  $a_1$  is  $z_1$  and sigmoid of  $a_2$  is  $z_2$ . So, these two put together we have at level 1 is  $g(z)$  at level 1 and this looks again remarkably same as what we had earlier.

So,  $a_1 = g(z_1)$ ,  $z_1 = w_1 a_0$ . So, exactly the same as before. So, all this is the forward propagation. Now, you might say, but that is not the forward propagation fully. What happens to the next step? So, the next step is going to be exactly the same in fact.

**(Refer Slide Time: 07:47)**

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$a_1 = g(z_1)$$

$$a_2 = g(z_2)$$

$$z^{(1)} = W^{(1)} a_0$$

$$a^{(1)} = g(z^{(1)})$$

$$z^{(2)} = W^{(2)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

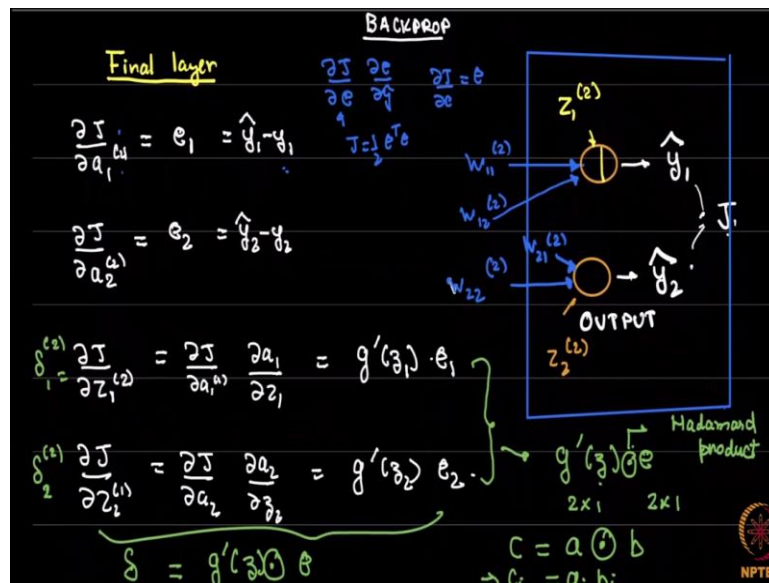
Now let us look at Back prop.

So, if we come here, we can now say  $z^{(2)} = w^{(2)} a^{(1)}$  and  $a^{(2)} = g(z^{(2)})$  nothing really changes. You see the outputs of this layer are related to these exactly the same as this is related to this. So, output here is  $a_1$ . So, you take  $a_0$  or  $x$  this goes multiplied by  $w$  gives  $z$  take  $g$  of  $z$  gives  $a_1$  multiplied by  $w$  gives  $z_2$ ,  $z_2$  take  $g$  and this gives you  $a_2$  and  $a_2$  was assumed to be the same as  $\hat{y}$ ;  $a_2$  has two components  $\hat{y}$  also has two components both these are 2 cross 1 matrices.

So, now let us look at back prop. Now that we have come to the end and we have found out  $\hat{y}$  is there. For multiple layers you will still keep on repeating this, just like we repeated in the chain as you can see there is really no difference between the chain in this case, but can we get

similar expressions for back prop just like we got easily for back prop in the scale or chain can we get it easily for this multi-layer case also and then let us go ahead.

(Refer Slide Time: 09:02)



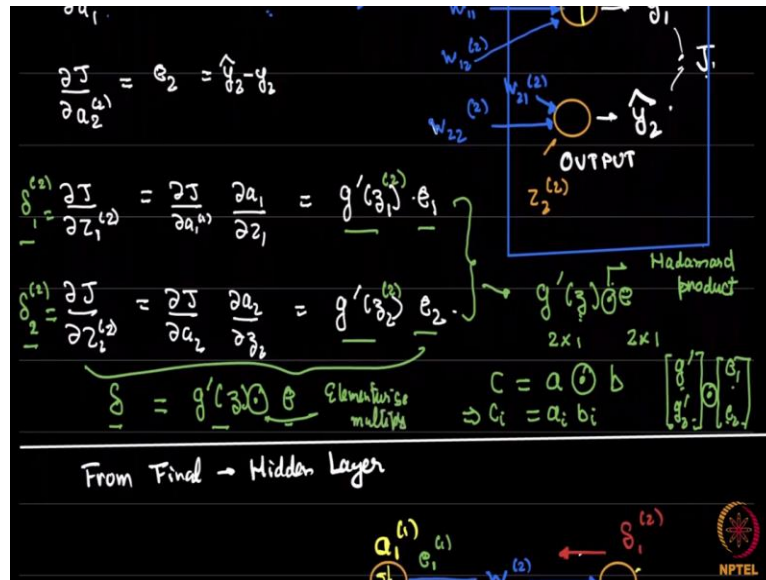
So, we are now starting with the back prop. So, as usual we will start with the final layer. So, I have taken this portion and I am just zooming in there. So, you will see that zoomed in portion here there is  $J, \hat{y}, \hat{y}_1, \hat{y}_2$  really speaking I should keep an  $e_1$  and an  $e_2$  here that is how you get the  $J$ , but that is okay I think you can imagine what is happening there already.

So, now we have inputs coming in from the previous layers. So, what are these inputs I seem to have cut off a few things here, but we can write that, that we write it in different color  $w_{22}^{(2)}, w_{11}^{(2)}, w_{12}^{(2)}$ . So, these are coming in into this layer, but just like last time we first calculate  $e_1$  and  $e_2$ . So,  $(\hat{y}_1 - y_1), (\hat{y}_2 - y_2)$ . Now  $y$  is  $\frac{\partial J}{\partial \hat{y}} e_1$  because actually speaking you should do  $\frac{\partial J}{\partial e} \frac{\partial e}{\partial \hat{y}}$  etcetera, if as long as we take  $J$  as  $\frac{1}{2} e^2 = \frac{1}{2} e^T$  you will always get  $\frac{\partial J}{\partial e}$  as back as  $e$ .

So, that is going to happen I have already shown you the matrix math in one of the earlier videos where we did the matrix derivatives etc. So, let us assume that we are taking in case let us say it was not a least square function it was something else, you can still differentiate it. So, the assumption here is you have found out  $\frac{\partial J}{\partial e_1}$  instead of writing it as  $\frac{\partial J}{\partial \hat{y}_1}$  and  $\hat{y}_2$  let me in fact write this as.

So,  $e_1, e_2$  you have got these errors here. Now what we need now is in fact I should be careful  $a_1$  and  $a_2$ . We are still with the final outputs of the final moment. Ignore that for now let us assume that once we have these errors, I want  $\frac{\partial J}{\partial z_1}$  here this again is not very clear this should be 2 and this should also be  $z_2^{(2)}$ .

(Refer Slide Time: 11:53)



Many of the times in this I have dropped the superscripts, but because we are kind of confusing. As long as you know where we are looking, we are looking at the final layer. So, this  $z_1$  and  $z_2$  refer to the second level linear activations. So, we come here  $\frac{\partial J}{\partial z_1}$  we defined that as  $\delta_1$  remember and  $\frac{\partial J}{\partial z_2}$  and that is defined as  $\delta_2$ , how do we calculate it?

Same thing as ever this is  $\frac{\partial J}{\partial a_1} \frac{\partial a_1}{\partial z_1}, \frac{\partial J}{\partial a_2} \frac{\partial a_2}{\partial z_2}$ . This is  $e_1$ , this is  $e_2$  which we just calculated here multiplied by  $\frac{\partial a_1}{\partial z_1}$  the relationship is  $g'$ . Just like what we did for the scalar chain nothing really changes here  $g$  of  $z$  is what relates  $z$  and  $a$ . So,  $g'(z_1), g'(z_2)$  if you want to be really, really specific this should be  $z_1$  at 2 and  $z_2$  at 2, but I have skipped that.

Now look at this relationship  $\delta_1 = g'(z_1)e_1, \delta_2 = g'(z_2)e_2$  you can write this in a compact notation as  $\delta = g'(z)e$ , but this multiplication is very specific because  $g'(z)$  here is  $\begin{bmatrix} g'_1 \\ g'_2 \end{bmatrix}$  and  $e$  here is  $\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$ . In the normal multiplication you cannot multiply these two matrices because they are not compatible both are 2 cross 1.

But we define a new multiplication called the Hadamard product this exists within the python library MATLAB; it is simply called dot star. So, if you want to do  $c$  is a vector which is the product of these two then it will be  $c_i = a_i b_i$  just like if I did  $c = a + b$  you would add element by element. This is called element by element multiplication also called element wise multiply.

Do not get lost in the notation all I am saying is  $\delta_1 = g'(z_1)e_1$  and  $\delta_2 = g'(z_2)e_2$  which you anyway knew. So, if all I want to do is go from here to here i multiply by a  $g$  prime. Now this expression again looks the same as it did in back prop in the chain. So, if you notice here, you will have delta is  $g$  prime into  $e$  at the same level. The same thing here except this was a normal multiplication because it was a scalar.

It is a dot wise multiplication or a element wise multiplication in this case, because it is a product a vector here and a vector here. Now we did this now, but from the final we wish to go to the hidden layer this one. So, we have these  $z$ 's in fact what we have now is  $\frac{\partial J}{\partial z_1}$  and  $\frac{\partial J}{\partial z_2}$  and what he desires now is simple we want  $\frac{\partial J}{\partial a_1}$  where  $a_1$  is  $a_1^{(1)}$  and  $\frac{\partial J}{\partial a_2}$ . If we do that then we can do the same trick here and keep on propagating.

**(Refer Slide Time: 15:25)**

The diagram shows a neural network with two input nodes  $z_1$  and  $z_2$  and two output nodes  $o_1$  and  $o_2$ . Weights  $w_{11}^{(2)}$ ,  $w_{12}^{(2)}$ ,  $w_{21}^{(2)}$ , and  $w_{22}^{(2)}$  connect the input nodes to the output nodes. Error terms  $\delta_1^{(2)}$  and  $\delta_2^{(2)}$  are associated with the output nodes. The total cost function  $J$  is indicated by a bracket on the right.

Handwritten equations for backpropagation:

$$e_1^{(1)} = \frac{\partial J}{\partial a_1^{(1)}} = \frac{\partial J}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial a_1^{(1)}} + \frac{\partial J}{\partial z_2^{(1)}} \frac{\partial z_2^{(1)}}{\partial a_1^{(1)}}$$

$$e_1^{(1)} = \delta_1^{(2)} w_{11}^{(2)} + \delta_2^{(2)} w_{21}^{(2)}$$

$$e_2^{(1)} = \frac{\partial J}{\partial a_2^{(1)}} = \frac{\partial J}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial a_2^{(1)}} + \frac{\partial J}{\partial z_2^{(1)}} \frac{\partial z_2^{(1)}}{\partial a_2^{(1)}}$$

$$e_2^{(1)} = \delta_1^{(2)} w_{12}^{(2)} + \delta_2^{(2)} w_{22}^{(2)}$$

So, we have  $\delta_1, \delta_2$  we want  $e_1$  and  $e_2$  let us remember that. So,  $e_1$  is  $\frac{\partial J}{\partial a_1}$  again ignore the superscript I will not keep on saying  $a_1^{(1)}$ . So,  $\frac{\partial J}{\partial a_1}$  now when I want  $\frac{\partial J}{\partial a_1}$ , I can get to this  $a_1$  this activation here from  $J$  in two ways either I come through this neuron and come here or I



can come through this neuron and come here. So, J is affected by this in two ways this path and this path.

So, we will accordingly write the chain rule  $\frac{\partial J}{\partial a_1}$  is  $\frac{\partial J}{\partial z_1} \frac{\partial z_1}{\partial a_1}$  or  $\frac{\partial J}{\partial z_2} \frac{\partial z_2}{\partial a_1}$ . So,  $\frac{\partial J}{\partial z_1} \frac{\partial z_1}{\partial a_1} + \frac{\partial J}{\partial z_2} \frac{\partial z_2}{\partial a_1}$ . why is this plus there? Because if I put up this a little bit one effect comes through this another effect comes through this and these two get added in J because these ultimately both add to contribute towards J because that is how derivatives work.

Once you understand this then we can write each one of these quantities what is  $\frac{\partial J}{\partial z_1}$  that is just  $\delta_1$ . So, we already have that. So, we have  $\delta_1$  what is  $\frac{\partial z_1}{\partial a_1}$ ?  $\frac{\partial z_1}{\partial a_1}$  is simply the weight connecting the two which is  $w_{11}^{(2)}$  of course, but I will just call it  $w_{11}$ . Similarly, what about  $\frac{\partial J}{\partial z_2}$  that is  $\delta_2$  by definition what is the weight connecting  $z_2$  and  $a_1$  that is  $w_{21}$ .

So, you have  $w_{21}^{(2)}$  so you add these two and you get  $e_1$ . So,  $e_1$  is  $\frac{\partial J}{\partial a_1}$  and that is just defined we just derived input. Similarly, if I look at  $\frac{\partial J}{\partial a_2}$  now  $a_2$  can come through  $z_1$  or through  $z_2$ . So, again  $\frac{\partial J}{\partial z_1} \frac{\partial z_1}{\partial a_2}$  or  $\frac{\partial J}{\partial z_2} \frac{\partial z_2}{\partial a_2}$ ;  $\frac{\partial J}{\partial z_1}$  is again  $\delta_1$ ,  $\frac{\partial z_1}{\partial a_2}$ . So,  $\frac{\partial z_1}{\partial a_2}$  is  $w_{12}$  and similarly  $\frac{\partial J}{\partial z_2}$  is  $\delta_2$  and  $\frac{\partial z_2}{\partial a_2}$  is simply  $w_{22}$ .

(Refer Slide Time: 18:03)

$$\begin{bmatrix} e_1^{(1)} \\ e_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} \\ w_{12}^{(2)} & w_{22}^{(2)} \end{bmatrix} \begin{bmatrix} \delta_1^{(2)} \\ \delta_2^{(2)} \end{bmatrix}$$

$$e^{(1)} = W^T(2) \delta^{(2)}$$

$$\delta^{(2)} = g'(z^{(2)}) \odot e^{(2)}$$

These expressions are almost exactly what we had in the scalar chain case! (Compare)

From Hidden - Weight Update

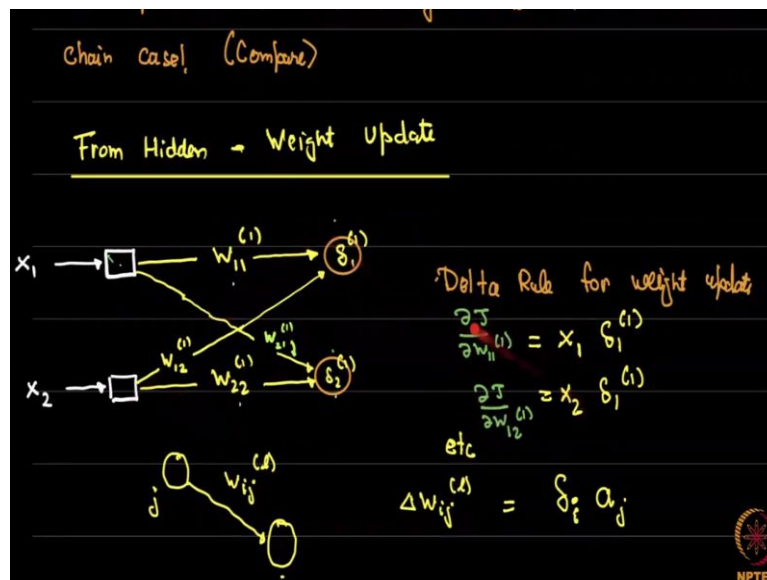
But what happens if we write this as a matrix something very neat happens here  $e_1 = w_{11}\delta_1 + w_{21}\delta_2$ ;  $e_2 = w_{12}\delta_1 + w_{22}\delta_2$ . If you notice this is exactly the  $w^{(2)}$  matrix, but it is transposed.



So,  $e_1$  is not  $w$  times delta, but it is  $w$  transpose times delta. Again, I will give you a comparison with what happens in the chain if you notice here  $e$  is  $w$  times delta all the only change that has happened here is  $e_{(k-1)} = w_k^T + \delta_k$  because now we are dealing with matrices.

So, this is also called the adjoint. So, what you have here what we have derived here is  $e_1$  is  $w$  transpose times delta 2. These expressions as we saw are almost exactly the same that we have in the scalar chain case as you can see you can simply compare with what I did in fact I did compare.

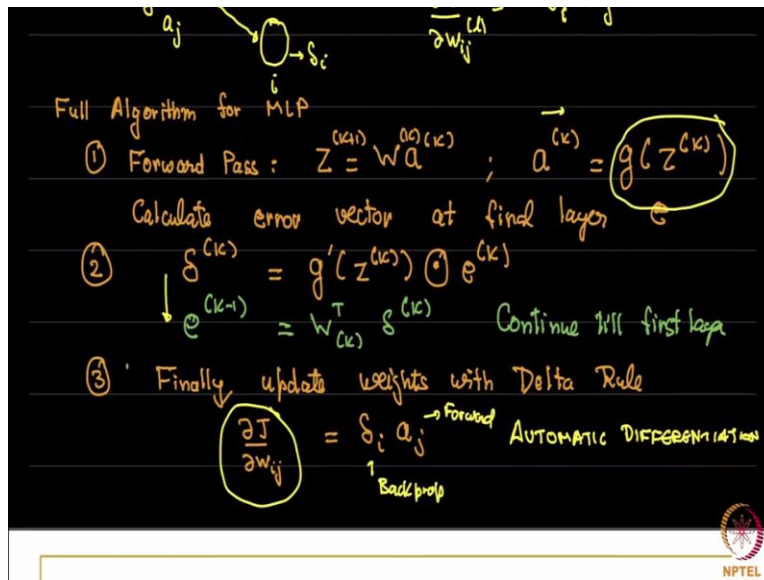
**(Refer Slide Time: 19:16)**



Now suppose I want the weight update. So, suppose I want  $\Delta w_1$  in fact instead of calling it  $\Delta$  let me call it  $\frac{\partial J}{\partial w_{11}}$ . Similarly, I am going to call this  $\frac{\partial J}{\partial w_{12}}$ . Now what is this if I want  $\frac{\partial J}{\partial w_{11}}$  I have to take the activation of the input which is  $x_1$  multiplied by the error in the output really speaking the delta of the output which is  $\delta_1^{(1)}$  that is it. If you have calculated  $\delta_1^{(1)}$  you can simply calculate  $\frac{\partial J}{\partial w_{11}}$  what about if you want  $\frac{\partial J}{\partial w_{12}}$ .

Then you do error sorry activation of the input which is  $x_2$  multiplied by  $\delta_1$  of the output. So, similarly you can actually calculate  $\frac{\partial J}{\partial w_{ij}}$  and simply as delta of the output  $\delta_i$  multiplied by  $a_j$ .

**(Refer Slide Time: 20:42)**



So, the full algorithm for the MLP is very straightforward it is almost exactly as what we did for the scalar chain and it is as follows. You first do the forward pass  $z$  is  $w$  times  $a$ . So, let me show you what is happening here, all we did in the forward pass is  $z$  is  $w$  times  $a$ . So, if it looks like this  $z$  is  $w$  times this activation, this activation is  $g$  of this, this  $z$  is  $w$  times this and this activation is  $g$  of that.

And you keep on going forward in case you have multiple layers more than that. So,  $z$  is  $w$   $a$ ,  $a$  is  $g$  of  $z$ . Calculate the error vector at the final layer then once again we do the same thing given an  $e$  find a delta. How to find the delta multiplied by  $g$  prime. Given a delta or given a delta let me show you the next figure probably that is a little bit clear given this delta find this  $e$  by multiplying with the transpose of this weight matrix.

So, just like forward multiplies by  $w$  inverse multiplies by  $w$  transpose. In fact, there is a very beautiful relationship between this and our general inverse methods. If time permits, I will discuss this in Week 12 when we come to some of the advanced topics or at least I give you an overview of advanced topics so that is it. Forward is  $z$   $a$ ,  $z$   $a$  so on and so forth inverses  $e$  delta,  $e$  delta and finally you just calculate once you know the  $a$  from the forward.


So,  $a$  is known from forward, all the deltas are known from the back. You can calculate the entire  $\frac{\partial J}{\partial w_{ij}}$ . Now notice you just did one forward pass to calculate all the  $a$ 's and you did just one back pass of course it is going through multiple layers, but the pass itself is one you go from the input to the or from the input to the output once and output to the input once and all the weights can be calculated at one shot.

So, this is why it is extremely clever as well as extremely cheap. So, this is the back prop algorithm. Now all through this algorithm we have been using this term  $g$  of  $z$  and this  $g$  of  $z$  we have assumed usually to be a sigmoid, but there are several common non linearities.

(Refer Slide Time: 23:27)

**Common non-linearities**

1.  $g(z) = \text{step}(z) = H(z)$ 
  - Graph: A step function that is 0 for  $z < 0$  and 1 for  $z > 0$ .
  - Annotation: "Not differentiable" at  $z = 0$ .
2.  $g(z) = \text{sgn}(z)$ 
  - Graph: A sign function that is -1 for  $z < 0$  and 1 for  $z > 0$ .
  - Annotation: "Not differentiable" at  $z = 0$ .
3. Sigmoid  $g(z) = \sigma(z)$ 
  - Graph: A smooth S-shaped curve from 0 to 1.
  - Annotations: "Differentiable", "Approx of H(z)", "Layer # of layer", "At  $z=0$   $\sigma(z)=0.5$ ", "poor  $\frac{\partial J}{\partial w}$ ".
  - Equation:  $g'(z) = \sigma(z)(1-\sigma(z)) = \sigma(1-\sigma)$
4.  $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ 
  - Graph: A smooth S-shaped curve from -1 to 1.
  - Annotations: "Common", "Approx of  $\text{sgn}(z)$ ", "slope of  $\tanh(z)$  at  $z=0$  is 1".
5. ReLU
  - Graph: A function that is 0 for  $z < 0$  and  $z$  for  $z > 0$ .
  - Equation:  $g(z) = \begin{cases} z & z > 0 \\ 0 & z < 0 \end{cases}$
  - Equation:  $\max(0, z)$



As I showed you historically first was this TLU or the threshold non-linearity, for example,  $g$  of  $z$  could be step function of  $z$ . So, the step function of  $z$  is if  $z$  is less than 0 it is 0 and if  $z$  is greater than 0 it is 1, so this just does this. There is another function called sign of  $z$ ,  $\text{sgn}$  of  $z$  which is 1 and  $-1$ . So, both these were used historically of course neither of these are differentiable.

So, these are both not differentiable and as you notice in our back prop algorithm, we have  $g$  prime  $z$  sitting there. So, these are not nice functions for two reasons, it is not just that these are not differentiable, but everywhere else their derivative is 0. So, basically this expression is going to look like at one point to infinity and everywhere else it is going to look like 0, so delta of  $k$  will as it is said it will not propagate.

So, this is not useful for back prop neither of these can be used with the back prop. The third one is the sigmoid function which we saw sigmoid of  $z$  essentially it is an approximation of  $h$  of  $z$ . So, as you can see it is a smooth step function except in the middle it goes to 0.5. This is differentiable  $\frac{1}{1+e^{-z}}$  and the derivative as you remember  $g'(z) = \sigma'(z) = \sigma(1-\sigma)$ .

This leads to a problem. So, the problem is like this at  $z$  equal to 0, sigmoid of  $z$  is of course 0.5, sigmoid prime of  $z$  is  $0.5 \times (1 - 0.5)$  which is 0.25. Now what is the problem here, the problem happens with large number of layers. When you have a large number of layers, each layer will get a multiplication by sigmoid or sigmoid prime. So, if I look at the chain each time I jump from here to here I get a  $g$  prime then another  $g$  prime then another  $g$  prime.

So, 0.25 into 0.25 into 0.25 as it becomes more and more layers it becomes very small and typically on a machine you can only represent till  $10^{-16}$ . So, this is bad for back prop. So, large number of layers means poor gradients, because of this  $g$  prime sitting there everywhere. Now you might say that is only here what about all these places if you notice everywhere else the gradient is actually lower it is 0.5.

And then it starts decreasing and you actually go to 0 here. So, here is where you get the maximum gradient. So, people had this clever idea of using tan h, tan h is sort of an approximation of sin of  $z$  goes from -1 to 1 and the slope here is approximately 1. So, almost never in practical networks do people use sigmoid they typically use tan h even if they want a simple non linearity.

Now since this slope is nearly 1 when you do  $g$  prime multiplied by  $g$  prime multiplied by  $g$  prime, you can actually propagate through a large number of layers. So, tan h does not easily disappear even if you have a large number of layers. So, in many networks for example what are known as RNNs tan h is fairly common even when you have a large number of layers like 100 layers or something of that sort 100 beat.

**(Refer Slide Time: 27:17)**



And derive it for far more complex networks, but it is being done automatically in framework such as tensor flow or even within MATLAB has what is known as auto grad automatic differentiation. In the next week we will apply these ideas and go back to the original problem as we saw neural networks themselves solve an inverse problem of finding the weights.

But once you have found the weights neural networks are a data model, how we use this data model with an inverse problem is something that we will see the next week. So, I will see you in Week 11. Thank you.