Social Network Analysis
Prof. Shivani Kumar
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

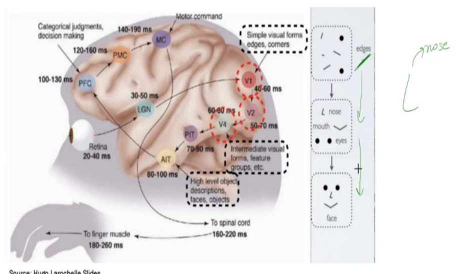
Chapter - 09
Lecture - 02

Hi, welcome back to the tutorial on deep learning. So, here last time we saw what is deep learning, how we can basically get started with deep learning. We saw the basic mathematical model proposed by McCulloch Pitts of the neuron. We saw the Rosenblatt's perceptron which is basically the newest perceptron, that is being used in this in the feed forward or the other deep learning architectures that we see today.

We also got a brief overview of how this perceptron learns. In today's class, we will dive deep into this learning algorithm. We will also see the feed forward network, that is the way a single perceptron or multiple perceptrons can be combined together to learn different and complex patterns, that may be present in the data. So, let us get started.

(Refer Slide Time: 01:25)

Feed forward network



Source: Hugo Larochelle Slides

So, first we will look into what is a feed forward network. So, we humans in our brain we have hundreds and thousands of neurons with each neuron performing some specific function. For instance, here if we have to identify an object that we are seeing. So, we want to identify whether the object is say whether it is a; it is a person, whether it is a cat or whether it is something entirely else. We look at the object through our eyes which basically acts as the stimuli which activates some neurons in our brain which then activates further different neurons in our brain.

And, with a combination of all these with the neurons, all the different neurons which are basically passing the information in a hierarchical fashion, we are able to make a judgment, we are able to make a decision about the object that is in front of us. So, here we can see that the object is being looked at by the eye and it is passed on the stimulus is passed on to the first neuron here which is then further passed on to the next neuron based on some kind of activation, that it does. Then, it is passed on to another, followed by another and another and so on.

Then, now each of this neuron in this whole pathway or this whole complex structure performs a specific different task. So, for instance it might be that the first few neurons, first few levels of neuron they just they are just identifying some sort of shapes in the object that we are seeing; that is where does the edges of the object lie, how many circles are there in the object, where does the circles lie right.

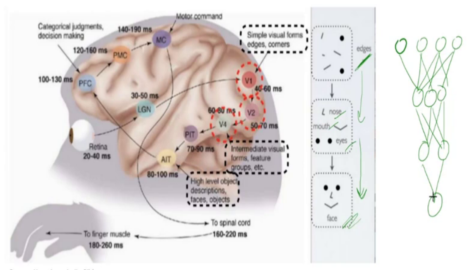
So, the first few preliminary neurons, first few levels of neurons they are just considered they are just considering these basic information's of finding edges and say the contours and the basic things, the total basic structure of the object in front of us. Then, the next few layer of neuron might be associating these object to say a larger object. For example, say we see we see a structure like this, this is being identified by the first few layers of the neurons.

The next few layers associate this structure which say a nose, if it is a person in front of us. So, the next few neurons it basically does a complex task of querying through our knowledge of what these structure might represents, might resembles to and then associate the structure to a already known, already seen object from our past.

So, the first few neurons as mentioned identify just the edges and the boundaries of the structure. Then, the next might be able to associate the structures to different objects and finally, the other the last few neuron, the higher level neurons which associate these smaller objects into a bigger image and can be; and can help us to identify whether these the combination of these structures make up a face or not. And, what kind of face does it make, is it an animals face or a humans face. Now, we so this all of this is working in a hierarchical fashion that is we have some.

(Refer Slide Time: 05:31)

Feed forward network



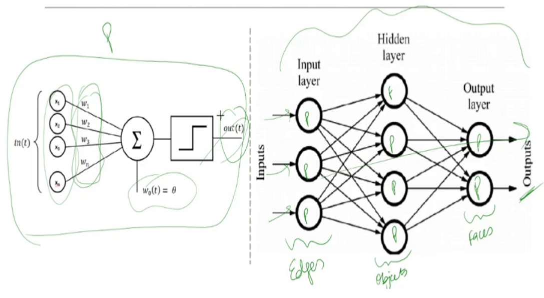
Source: Hugo Larochelle Slides

So, here for instance we might have some neurons here which identify the edges and the information of these neurons they are passed on to the next neurons which identifies say the different structures of nose, mouth and eyes. And, these are further or further passed on to the next level of neuron which identifies whether it is a face or not. And, then lastly it identifies of which what kind of face is it.

Now, this is something that we do on a daily basis without even realizing it. But, in order to mimic this you this behavior in computer machines, we are proposing to use the basic perceptron in an hierarchical fashion.

(Refer Slide Time: 06:20)

Feed forward network



So, here we have this perceptron here which is a single unit which takes the input that we just mentioned as a real numbers. It has some face associated with each of these inputs and it performs some kind of aggregation over these inputs to give us an output which basically says if you want to activate this perceptron or not.

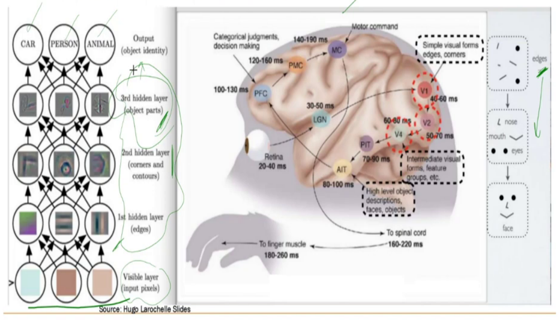
Now, we might use multiple of such perceptron. So, say this perceptron we let us call it P, we might use multiple of such perceptrons in our network and each of this perceptron is responsible for identifying different things.

For instance, we might say that these first few perceptrons, they are to identify just the edges of the image that we are seeing. Then, this hidden layer it is responsible for associating these edges to objects that is whether it is a nose or a mouth or an ear and such that. Then finally, this output layer it is responsible for determining whether these objects together constitutes a face or not. And then finally, these outputs are given out to the environment and we make a decision based on the outputs that we receive here.

So, if we are to employ the perceptron in such a way. So, basically this structure where we use multiple perceptrons in a hierarchical fashion, it this is known as a feed forward network or multilayer perceptrons. So, as the name suggests there are multiple layers of the perceptron architecture. So, these are known as multilayer perceptrons or a feed forward network, since we are feeding the input in the forward fashion and there is no feedback that is involved here right. So, feed forward network or the multilayer perceptrons.

(Refer Slide Time: 08:22)

Feed forward network


CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)
Categorical judgments, decision making
Motor command
Simple visual forms edges, corners
Intermediate visual forms, feature groups, etc.
High level object descriptions, faces, objects
Retina 20-40 ms
To spinal cord 160-220 ms
To finger muscle 180-260 ms
edges
nose
mouth
eyes
face
Source: Hugo Larochelle Slides

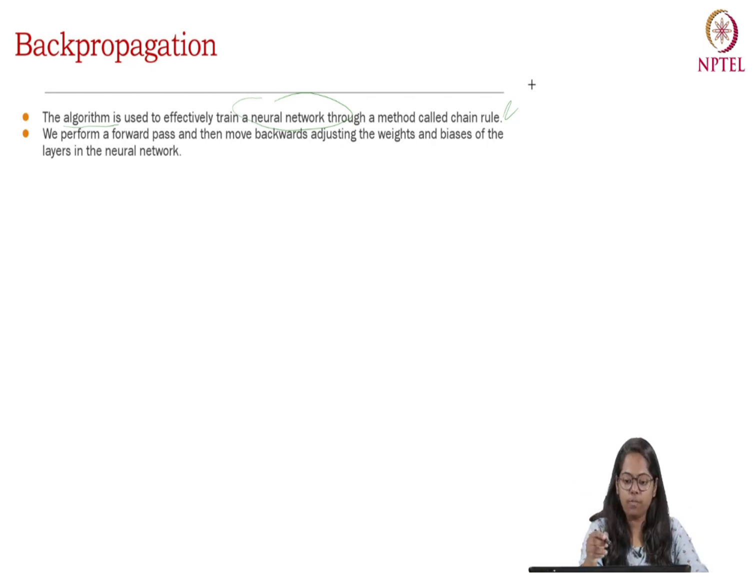
Now, as we saw before just like in our brains first few neurons, first few stimulus of our brain they are responsible to identify the visuals that we are seeing, the colors that we are seeing might be. Then, the first hidden state, they are just they are just identifying the edges in the image that we are seeing. The second hidden state is identifying the corners and the contours so, that we are able to differentiate the different objects in the image, that how many objects are there and how they are different to each other.

Then, the third layer it basically is associating these different edges and contours into constituting an object from the from this layer. And so, as saw here as we saw here, the first layer identify the edges and then we are associating these edges into making these objects such as nose, mouth and eyes. So, this is what is happening here in the third layer. And then finally, we identify the object in the final layers of our architecture where we can say where we can identify whether the structure is a car, it is a person or an animal right.

So, yeah so, this is how a feed forward network works. And why do we need it? Because, a single perceptron or a single neuron might not be able to capture all the nitty-gritties of the tasks that you want to achieve. So, for here we want to we want to achieve the task of object identification. So, a single perceptron might not be able to capture all the different kinds of attribute that we want to consider in order to identify the object.

Whereas, if we employ various neurons across different layers, we might able to capture this object identity in a better fashion. As multiple neurons across multiple layers will be capturing different attributes, different properties of the image that we are seeing and then must be learning different or different associations between these objects of the image, in order to generate the better output or for object identity.

(Refer Slide Time: 10:46)



Backpropagation

- The algorithm is used to effectively train a neural network through a method called chain rule.
- We perform a forward pass and then move backwards adjusting the weights and biases of the layers in the neural network.

Now, we know we have seen what is a feed forward network. Now, in this feed forward network as we have seen before that each perceptron it is associated with some kinds of weights and bias and based on these weights and bias it gives us an output correct. But, now if we are to learn, if we are to modify these weights and bias in a way so, that the predicted output of our neuron, the predicted output of our whole architecture is the output that we want is the desired output.

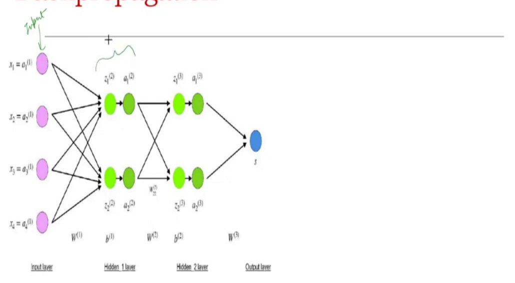
Then, we must learn these weights and bias in a way that whatever computation that is happening in this network gives us an output that is closest to the desired output. So, it can be this whole learning process, it can be achieved through a process known as back propagation. So, as the name suggest back propagation means propagating the error that we achieve as the output layer of the network to the throughout the network till the input layer.

Since, we are back propagating this error and based on this error, we are modifying the values in a backward fashion; this whole process it is called as back propagation. So, this is basically an algorithm which is used to train a neural network. And, the method that it uses to identify the amount of change that you want in the weights and the bias of our network, that method is called as chain rule. We will see this right now.

So, the first process before back propagation, it is to definitely do a forward pass through the network. That is firstly, we initialize these weights and bias in a fashion like there are multiple ways for initializing these weights and bias. One of them can be to just do random initialization, that is we randomly set the bias and weights for the initial level and just perform a forward pass in the network.

(Refer Slide Time: 13:18)

Backpropagation

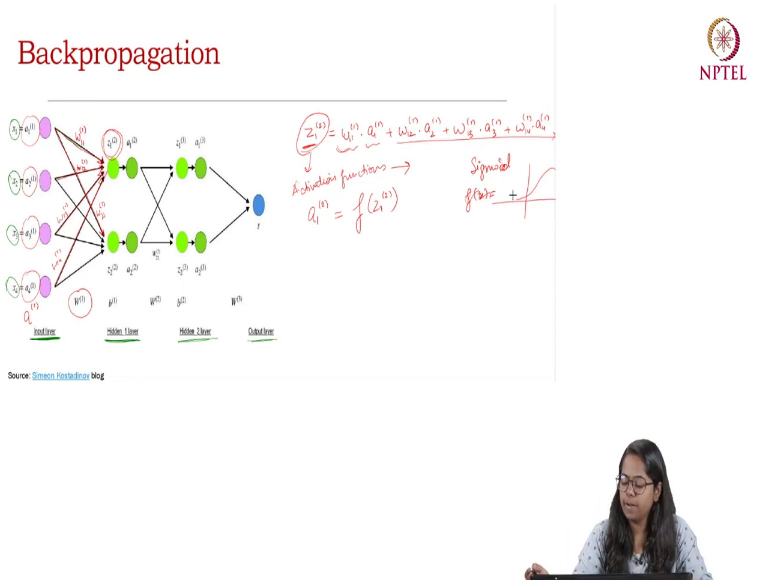


Source: Simeon Kostadinov blog

So, we will just look at what a forward pass is, but before we do that let us take an instance of this, let us suppose that we have this kind of a feed forward network with us. So, basically it has 4 layers, this is the input layer right. Then, we have the first hidden layer here right, here it is actually written. We have the this is the input layer, then we have the first hidden layer here, then we have another hidden layer here and the which is followed by the output layer.

So, we basically have 2 hidden layers, an input layer and an output layer in an our network. The input layer, it takes the input from it take the real valued inputs from whatever source that we have. Then, this hidden layer, it calculates the weighted aggregated sum of the inputs that we receive here. So, basically this line that you can see here this line, it is this edge is basically associated with the weight that we want to give to this input.

(Refer Slide Time: 14:36)


Backpropagation

$z_1^{(2)} = w_{11}^{(1)} \cdot a_1^{(1)} + w_{12}^{(1)} \cdot a_2^{(1)} + w_{13}^{(1)} \cdot a_3^{(1)} + w_{14}^{(1)} \cdot a_4^{(1)}$
Activation functions →
$a_1^{(2)} = f(z_1^{(2)})$
Sigmoid
$f(x) =$
Input layer
Hidden 1 layer
Hidden 2 layer
Output layer
NPTEL

Source: Simeon Kostadinov blog

So, we can call it w 11 and since it is of the first layer then it will be then there will be a superscript of 1 here. This is just like the norm in which the weights and the biases and the other values of the network are written. Then, we have this another edge associated with this first input neuron which basically tells us how much weight do we have to give to this neuron for the second neuron of the first hidden layer.

So, this weight can be written as w say 21, that is for the second neuron of the hidden layer and the weight is associated with the first input. And, again since this is the first layer, then a superscript of 1 right. Then, this particular weight it can be written as w 12 and 1 that is it is associated with the first neuron of the first hidden layer and the second input right.

So, based on these values and similarly this will be w 13 and this will be w 1 4 of the first layer right. Now, we have this input layer which contains a 11, a 12, a 1, a 31 and a 41 which together can be written as a superscript 1, that is all the values of a present in the first layer. Then, we have all these weights for the between the first layer and the first hidden layer which can be represented as w superscript 1.

Then, we have these resultant values for z, the resultant value for the that is the weighted aggregate. So, here if we just look at this first z here so, z 1 of the second layer it is basically the weighted sum. So, it is like if we multiply w 11 of the first layer with a 1 of the first layer plus w 12 of the first layer into a 2 of the first layer plus w 13 of the first layer into a 3 of the first layer and finally, w 14 of the first layer into a 4 of the first layer right.

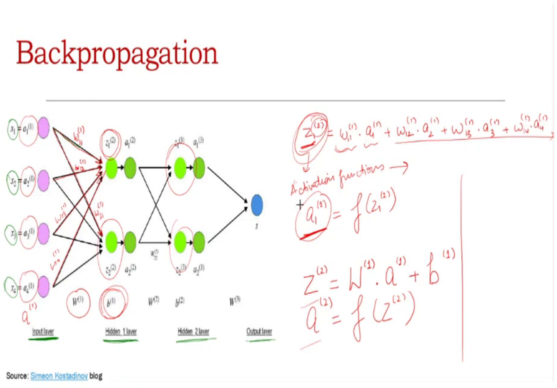
This gives us the value of z 1 of the second layer. Now, based on this value, since it since this is a real number, this is a real number, all of these are real numbers; we would get a real number for this z 1. But, now since it is a perceptron, we might want the value for the like the output value for this perceptron to be a Boolean variable. So, we have multiple activation function, something that is called as activation functions which basically modifies which is basically a transformation of this aggregated sum into a into an activated value.

So, there is so, we will not go into the depth of the activation function, but the only thing that is important here to know is that it is basically a transformation, a non-linear transformation from the input whatever comes to this activation function to the output right. So, this a value that we are getting, this a 1 for the second layer its basically after we are passing this z value through an activation function right.

So, there is one activation function known as sigmoid which like if we have to calculate the sigmoid for f x, then it will be something the graph of the value should would be something like this, that is for the 0th value we will be getting some negative values and for the positive values, for the one values we will be getting some positive values right. So, this is just one kind of activation function. But, other than that we have many different activation functions.

And, this is how we basically get the value for a, that is we pass the aggregated sum value, the weighted sum through an activation function. So, this is how we get the value of a. Now, if you so, we see that how do we get a, how do we get z. Now, for the whole hidden layer 2, we can write it without this subscript because the superscript basically represents the layer.

(Refer Slide Time: 19:56)



Backpropagation

$$z_1^{(2)} = w_{11}^{(1)} \cdot a_1^{(1)} + w_{12}^{(1)} \cdot a_2^{(1)} + w_{13}^{(1)} \cdot a_3^{(1)} + w_{14}^{(1)} \cdot a_4^{(1)}$$

Activation functions $\rightarrow$

$$a_1^{(2)} = f(z_1^{(2)})$$

$$Z^{(2)} = W^{(1)} \cdot a^{(1)} + b^{(1)}$$

$$a^{(2)} = f(Z^{(2)})$$

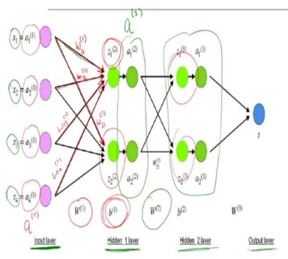
Source: Simeon Kostadinov blog

So, for the whole hidden layer 2, we can write that z of hidden layer 2 is actually equal to the w, the weights associated with the hidden layer 1 into this a's that are associated with the hidden layer 1 plus; now we this hidden layer basically there is a bias here which we need to add to this. So, that we can compare the whole aggregated sum with whether either it is greater than 0 or not right. So, plus b of the first layer.

Now, if this z 2 and then again this a 1, a 2 here will be the function will be the value of this z 2 after it is passed through the through an activation function right. So, now, we see that this is how we get the z 2 and the a 2 right. So, and similarly with the same approach we can get the values for z 22 to then the for the second hidden layer also, we can get the value like this. So, for the second hidden layer, it will be like we will have so, let us write it down here.

(Refer Slide Time: 21:21)

Backpropagation



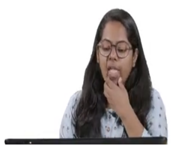
$Z^{(3)} = W^{(2)} \cdot a^{(2)} + b^{(2)}$

$a^{(3)} = f(Z^{(3)})$

$Z^{(2)} = W^{(1)} \cdot a^{(1)} + b^{(1)}$

$a^{(2)} = f(Z^{(2)})$

Source: Simeon Kostadinov blog

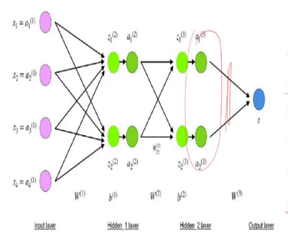


We will have for the this second hidden layer, we will have z of 3 equal to the w 2 that we have here, the weights associated with layer 2 into this a i's value that we are getting; since this is the input for our second layer.

So, we can write them down as a of superscript 2, since it is associated with the second layer. Then, w weights of the second layer multiplied by the activation of the second layer and then plus the bias of the second layer. And finally, this a 3 that we have is the activation function applied to the z value for the layer 3 right.

(Refer Slide Time: 22:11)

Backpropagation- Forward pass



$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at Hidden$_1$ layer*

$a^{(2)} = f(z^{(2)})$ *activation value at Hidden$_1$ layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at Hidden$_2$ layer*

$a^{(3)} = f(z^{(3)})$ *activation value at Hidden$_2$ layer*

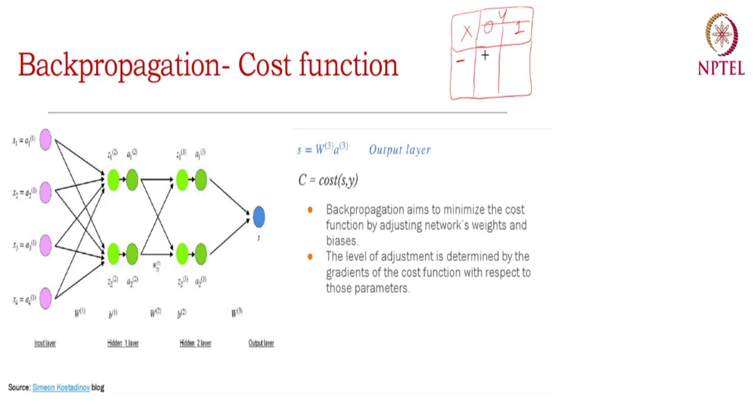
$s = W^{(3)}a^{(3)}$ *Output layer*

Source: Simeon Kostadinov blog

So, we just see the equations here. We have the input layer that is x that is the first a's that we have which is equal to the x. Then, we have the hidden layer 1 for the hidden layer 1. We have an equation like as we just saw previously z 2 is equal to the weights associated with the first between the weights associated between the first layer and the first hidden layer multiplied by the input plus the bias.

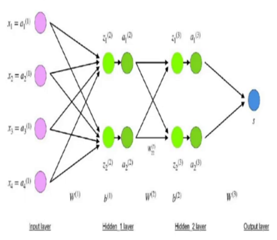
And, then we have the activation function that is applied to the hidden layer 1 and similarly we have the z at hidden layer hidden layer 2 and the activated value for the hidden layer 2. Now finally, at the output layer, we have s which is equal to the weights associated with the weighted sum of the values that we are getting at the last hidden layer and the corresponding weights right. So, the final s is W 3 that is the weights for hidden layer 3 for the total layer like the third layer and the activated value of the third layer right.

(Refer Slide Time: 23:18)



Backpropagation- Cost function

X	0	Y 1
-	+	



$s = W^{(3)}a^{(3)}$ *Output layer*

$C = cost(s,y)$

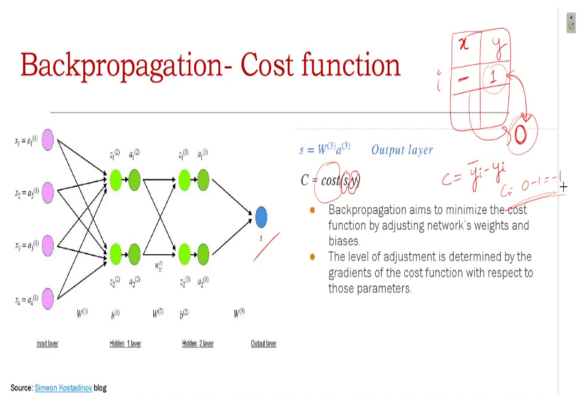
- Backpropagation aims to minimize the cost function by adjusting network's weights and biases.
- The level of adjustment is determined by the gradients of the cost function with respect to those parameters.

Source: Simeon Kostadinov blog

Then, now this s that we have, the output that we are getting here; it is not necessary that this output is the same as the desired output that we want. So, for instance suppose we are to perform a task of sentiment analysis at as we discussed in the last class. So, for the task of sentiment analysis, there are basically two classes with us, that is positive sentiment and a negative sentiment.

So, for instance 0 represents the negative sentiments and 1 represents the positive sentiment. Now, this these are basically the value that Y can take and we have some input X with us. So, now, for instance for a particular X our Y is so, let us create this table in a different fashion.

(Refer Slide Time: 24:16)



Backpropagation- Cost function

$s = W^{(3)}a^{(3)}$ *Output layer*

$C = cost(s,y)$

- Backpropagation aims to minimize the cost function by adjusting network's weights and biases.
- The level of adjustment is determined by the gradients of the cost function with respect to those parameters.

$c = \bar{y}_i - y_i$

$c = 0 - 1 = -1$

Source: Simeon Kostadinov blog

Suppose, we are given this x and y as the as our desired data and for a particular value for x's the ground truth label of y that is the actual sentiment associated with this x is basically positive 1. However, our model whatever model that we have with say random initialization of weights, it is suggesting that this input x its actually belonging to a negative sentiment which is not true in this case which is basically a wrong prediction.

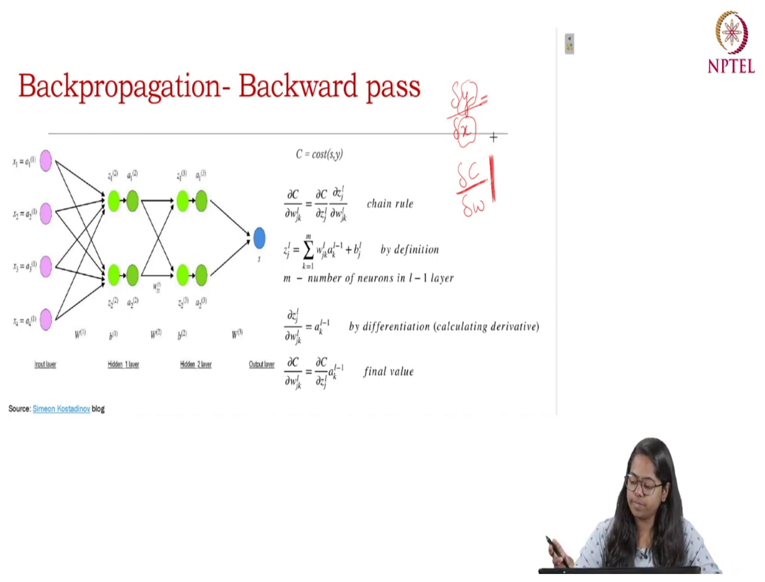
So, we want to modify a network in such a way so, that the model is able to learn that it is predicting it wrongly and it modifies its weight and biases such that for the future prediction it becomes correct right. So, in order to do that, the first thing first thing that we need to identify is whether it is predicting it correctly or not. So, to check whether it is predicting correctly and to check how wrong the network is performing, there is something that is called as a cost function right.

Now, again we will be not going into the depth of different kinds of cost function that are present. But, there are just like an activation function, there are different cost functions that are present in the whole deep learning paradigm. So, here we will just learn about the intuition behind this cost function. So, basically this cost function it is calculating how the desired, how the predicted output s is different from the desired output y.

So, for instance here a cost function can simply be the difference between the predicted value and the desired value. So, here for the case of sentiment analysis cost functions can be simply like y i bar y i minus y i, where y i bar is the suppose this is the ith label right. So, y i bar is the predicted value of y that we are getting which is either 0 or 1 and y i is the ground truth value that we have. So, for example, if with the predicted value 0 and the ground truth is 1; so, here the cost would be 0 minus 1 equal to minus 1.

Now, now we know that there is something called cost and we know that we can gauge the difference or the basically the issues, the issues that our model has in order to identify the correct output. So, now we want to use this cost, we want to use this particular value in order to modify the weights and bias. So, that this cost is minimized, that is this difference between the predicted value and the actual value is the least, is it is minimized that the least that we want right.

(Refer Slide Time: 27:38)



Backpropagation- Backward pass

$C = cost(s,y)$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \quad \textit{chain rule}$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l \quad \textit{by definition}$$

m – number of neurons in $l-1$ layer

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \quad \textit{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} \quad \textit{final value}$$

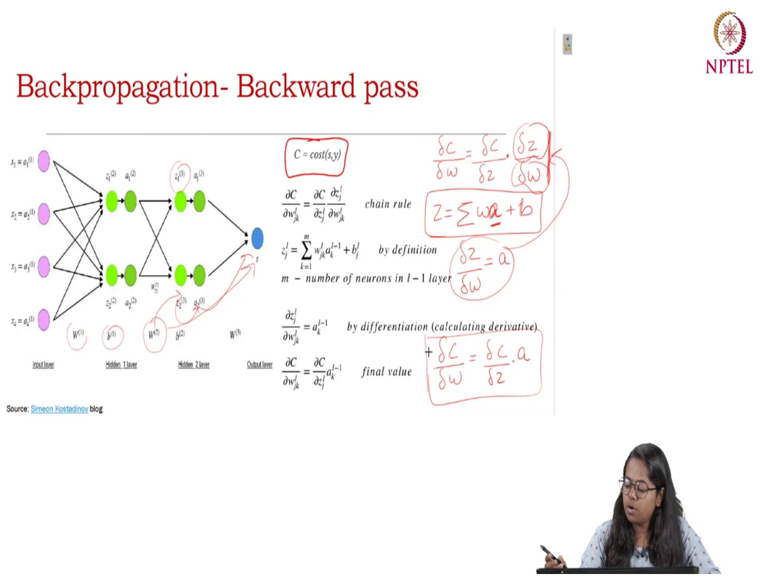
Source: Simeon Kostadinov blog

So, now, to do that we will calculate the derivatives. Now, why the derivatives? Because so, we know like what a derivative is, suppose we have to calculate the derivative of y with respect to x; then what does this derivative signify? It signifies that with the change in the value of x, how does the value of y change correct? So, we want to see that with how much change in the value of weights right, how does this value of cost change right?

So, if we are to change this value of weights towards a particular direction, how will the value of cost change in such a scenario in which direction, will it go will it increase or decrease? So, based on the derivative, we know that we want to minimize this cost. So, that is why we will based on the change that we observe here, based on the value that we observe here, that how much the C will change based on base; we will take this C in a direction such that it is minimized.

Now, in order to see how much the weights change, we need to calculate this derivative correct. Now, how to do that right? So, this is done by something called as chain rule. Now, we have the cost here, the C here that is the cost between the predicted value and the actual value. Then, we need to see how does the weights and biases are affected by this C.

(Refer Slide Time: 29:35)



Backpropagation- Backward pass

$\boxed{C = cost(s,y)}$

$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j}\frac{\partial z^l_j}{\partial w^l_{jk}}$ chain rule

$z^l_j = \sum_{k=1}^{m} w^l_{jk}a^{l-1}_k + b^l_j$ by definition

m – number of neurons in $l-1$ layer

$\frac{\partial z^l_j}{\partial w^l_{jk}} = a^{l-1}_k$ by differentiation (calculating derivative)

$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j}a^{l-1}_k$ final value

$\frac{\delta C}{\delta w} = \frac{\delta C}{\delta z} \cdot \frac{\delta z}{\delta w}$

$\boxed{Z = \sum w\underline{a} + b}$

$\frac{\delta z}{\delta w} = a$

$+\boxed{\frac{\delta C}{\delta w} = \frac{\delta C}{\delta z} \cdot a}$

Source: Simeon Kostadinov blog

So, that is we need to calculate the change in C based on the change in weight. Now, this can be done by using chain rule, that is we calculate this derivative of C based on say the z value that we have and, then the change in this z value based on the weights. Since, we see here that this weights here, they are not directly interacting with the with the cost C or the output s, rather that interacting with this z and this z is further with a and this a further with s.

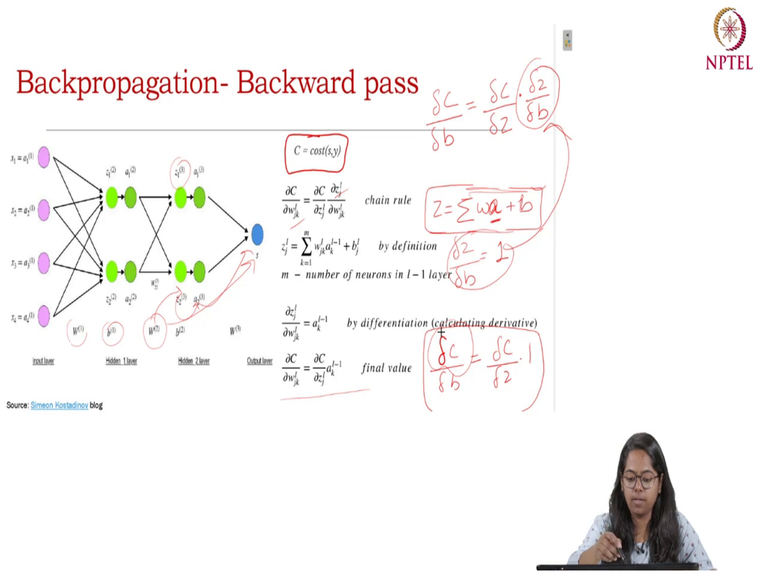
So, we kind of move backwards, one step at a time. We see that how is this s affected by this z, sorry how is this s affected by this z and further how is this z affected by the weights. So, to do that we to calculate the amount of change in the cost with respect to the weights, we first calculate the amount of change in this cost with respect to the z and the amount of change in z with respect to the weights. So, this is basically the chain rule. And, if we need to move more backwards, we will do we will that, we will we will just see further.

Now, here we know that this z, this z is given by the equation of the summation of all the w's, the weight weighted sum right; the w's multiplied by the weights plus the bias right. So, in this particular; in this particular representation that I am writing, I am just skipping the subscript on the superscripts. So, that it becomes easier to understand here and then we can add the superscription subscript based on the layer and the value for the z that we are working on right.

So, it is more like a generalized form. Now, since we know that this is what z is we can calculate this derivative right. So, the change in z with respect to w, we can calculate this derivative and we see that it is basically a right. The derivative or based on this formula and with respect to w is a. So, we replace this value here and we see that the change in C with respect to w can be given as the change in C respect to z multiplied by the a correct.

So, now this is basically the change in the cost function based on the particular weights. Now, similarly based on this z value, I am not erasing that z value right now. So, this we calculated for the weights. Now, we also want to modify the bias, in order to minimize the cost function.

(Refer Slide Time: 32:42)



Backpropagation- Backward pass

$$\frac{\delta C}{\delta b} = \frac{\delta C}{\delta z} \cdot \frac{\delta z}{\delta b}$$

$$\boxed{C = cost(s,y)}$$

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j}\frac{\partial z^l_j}{\partial w^l_{jk}}$$ chain rule

$$\boxed{z = \sum wa + b}$$

$$z^l_j = \sum_{k=1}^{m} w^l_{jk}a^{l-1}_k + b^l_j$$ by definition

$$\frac{\delta z}{\delta b} = 1$$

m – number of neurons in $l-1$ layer

$$\frac{\partial z^l_j}{\partial w^l_{jk}} = a^{l-1}_k$$ by differentiation (calculating derivative)

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j}a^{l-1}_k$$ final value

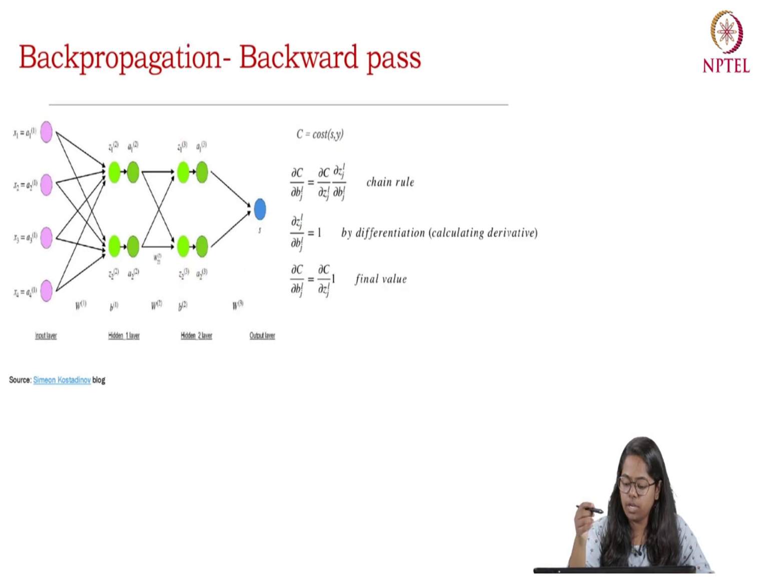
$$\frac{\delta C}{\delta b} = \frac{\delta C}{\delta z} \cdot 1$$

Source: Simeon Kostadinov blog

So, we also need to calculate change of cost function with respect to the bias. So, it can again be used by the by the chain rule that is the change in cost with respect to z and the change in z with respect to the bias. Now, this value based on the value of the equation of z that we have it can be calculated. So, derivative of z with respect to b would be basically 1 right.

So, we replace this value here and we see that derivative of C with respect to the bias is actually the derivative of C with respect to z into 1 correct. Now, we just see, we just add this is more like in a general fashion, we add the superscripts and the subscripts here to just identify that for each layer and for each of the pair of weights, this is how we can calculate the change of change of the cost with respect to the weights and the bias.

(Refer Slide Time: 33:48)



Backpropagation- Backward pass

$C = cost(s,y)$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \quad \textit{chain rule}$$

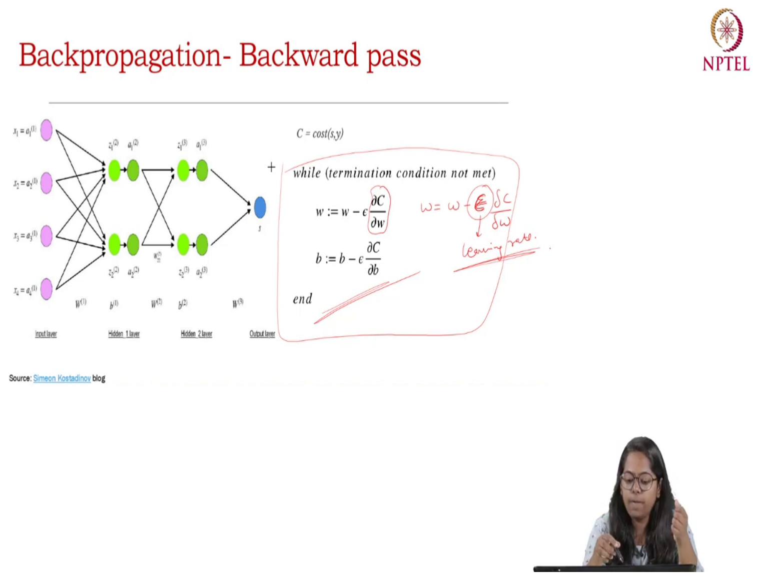
$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \quad \textit{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}1 \quad \textit{final value}$$

Source: Simeon Kostadinov blog

So, the next is basically the bias that we are the change bias that is how it is affecting.

(Refer Slide Time: 33:55)



Backpropagation- Backward pass

$C = cost(s,y)$

```
while (termination condition not met)
    $w := w - \epsilon \boxed{\frac{\partial C}{\partial w}}$
    $b := b - \epsilon \frac{\partial C}{\partial b}$
end
```

$w = w - \underset{\text{learning rate.}}{\boxed{\epsilon}} \frac{\delta C}{\delta w}$

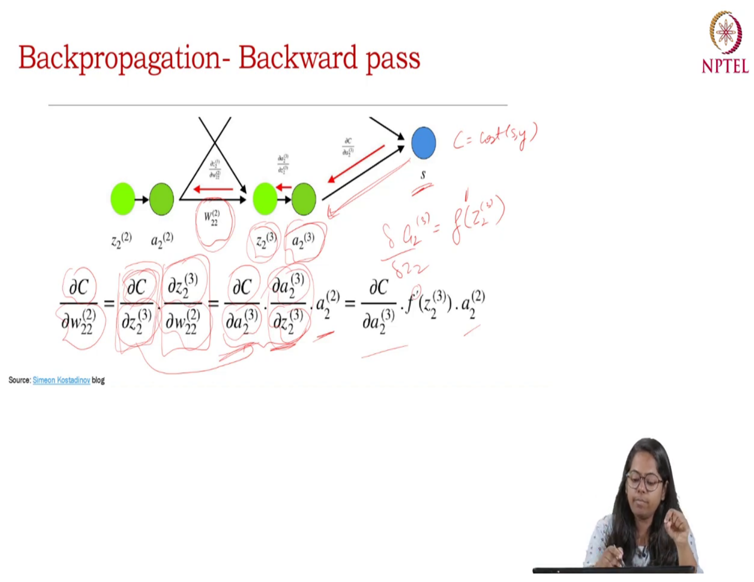
Source: Simeon Kostadinov blog

And finally, what we do is that based on this change, based on this derivative, based on the direction of the change; we modify our weights for the next iteration. So, we modify our weights in such a way that we multiply this derivative, this change that we want in the weights value with a learning rate here. So, this learning rate is something which regulates that at a particular iteration how much like the what would be the amount that you want this weights and a bias to mode to be modified.

So, this learning rate is basically something that helps us to regulate the convergence of the solution, that is we can say that the solution has converged when the when the change in the weights and bias between two iterations is below a threshold, that is we can say that the weights and bias in between the two iterations are more or less similar. So, that is when it can be one of the criteria for convergence. So, that is when we can say that the model is converged and this learning rate helps us to regulate the say the rate of that convergence.

So, it is essential to set this learning rate to a correct value, since a higher learning rate or a lower learning rate from the say the optimal value can result in say haywire learning. So, again not going into the detail of it, not going to the depth, since it is more like an overview of all the different architectures and this structure of deep learning. So, not going into the detail of learning rate, but yeah it is something which regulates the amount and the extent of convergence in our model.

(Refer Slide Time: 35:54)


NPTEL
$z_2^{(2)}$
$a_2^{(2)}$
$W_{22}^{(2)}$
$\frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}}$
$z_2^{(3)}$
$a_2^{(3)}$
$\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}}$
$\frac{\partial C}{\partial a_2^{(3)}}$
s
$C = \text{cost}(s, y)$
$\frac{\delta a_2^{(3)}}{\delta z_2} = f'(z_2^{(3)})$

Backpropagation- Backward pass

$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \cdot a_2^{(2)} = \frac{\partial C}{\partial a_2^{(3)}} \cdot f'(z_2^{(3)}) \cdot a_2^{(2)}$$

Source: Simeon Kostadinov blog

Then so, this is an example of the forward pass, here we have the s the cost here is just like the cost between the s and the actual y. Now, we want to calculate the how to change this value of w 22, this weights based on the x of based on this cost. So, we calculate the derivative of this cost based on these weights and in order to do that, we apply the chain rule. We have this cost and we see here we move backwards here.

So, we see that first this s is this cost is interacting with a 2. So, we just and this a 2 is basically coming from z 2. So, we have this derivative of cost with z 2 and then z 2 with w 22 correct, since this z 2 is directly interacting with w 22. So, we have this value of z 2 with w 22. Then, we have this now again this whole thing can be can be written in with the help of this a 2, since we see that cost is the one directly interacting with a 2.

So, we again use apply chain rule here and write this particular thing like this, that the derivative of C with respect to a 2; the red multiplied with the derivative of a 2 with respect to z 2. And, as we saw before this value that is z with respect to weight, we saw this here that is derivative with respect to weight is basically this value, that is a derivative multiplied by a. So, we have this here the derivative multiplied by a.

Then, again to calculate derivative of this. So, what is a 2? a 2 3 is basically the activation function applied on z 2 3 and the derivative of this would be the derivative of this function. So, we have this f derivative here and the normal cost of the cost, the derivative of cost with respect to a 2 right. So, this is how the backward pass takes place and based on this particular small algorithm, we modify the weights and biases with respect to these derivatives in order to get like in order to achieve convergence and to get the optimal value for these weights and biases.

Now, today we saw a feed forward neural network and we saw the major algorithm that is being used to optimize these neural networks. So, that learning actually happen and we get the optimal value for this weights and biases; so, that we get like a better accuracy, better performance for our models. Now, in the next class we will see different architectures other than this feed forward network. For example, convolutional neural networks and recurrent neural networks and then we will look into something called attention as well.

So, thank you.