

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

Module - 4.4
Lecture - 04
Backpropagation (Intuition)

So, we are in module 4.4 of this longest lecture on Backpropagation and feed forward neural networks. So, we introduced a feed forward neural networks, we saw the input layer hidden layer and the output layers. And we saw that the output layer actually the output function depends on the task at hand. And we considered two main tasks, one was classification the other was a regression.

For regression it made sense to use a linear layer at the output. Because we did not want the outputs to be bounded, they could be any range. And for the classification problem we realized that we want a special kind of output, because we are looking for a probability distribution over the output. And for that we use the softmax function. And in both cases we used a different kind of a loss. For the regression problem the squared error loss made sense, because we predict some values and we want to see how far we are from those values.

But for the other case the classification we realize that it is a distribution, so maybe we could use something; which allows us to capture the difference between the true distribution and the predicted distribution.

(Refer Slide Time: 01:20)

		Outputs	
		Real Values	Probabilities
Output Activation	Linear	Softmax	
Loss Function	Squared Error	Cross Entropy	

$J(\theta)$
0.9

- Of course, there could be other loss functions depending on the problem at hand but the two loss functions that we just saw are encountered very often
- For the rest of this lecture we will focus on the case where the output activation is a softmax function and the loss function is cross entropy

Mitesh M. Khapra CS7015 (Deep Learning): Lecture 4 22/57

And therefore, we had this figure emerging which was depending on the output, whether it is real values or probabilities. You will have different types of output activation functions and different types of losses.

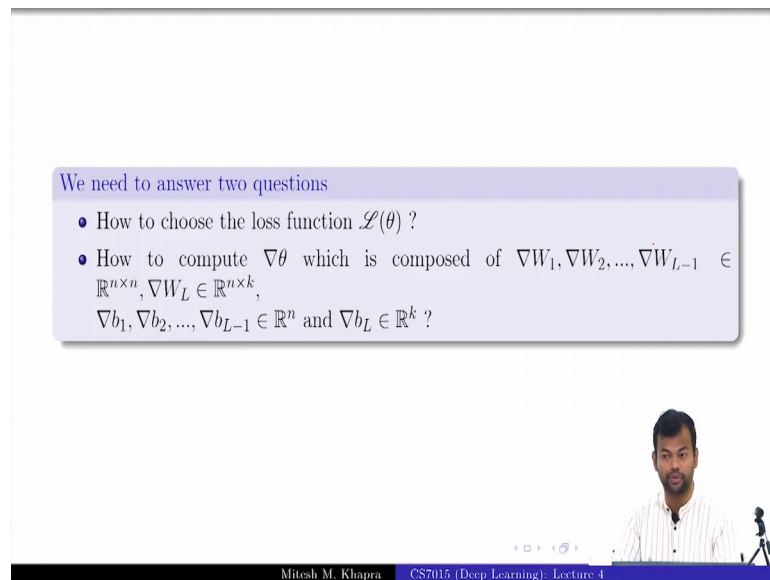
And of these combinations today we are going to focus on softmax and cross entropy, and our aim is to actually find these gradients, right? Remember there are many of those we have seen this large matrix which had many such partial derivatives, and we want to find that entire matrix. I hopefully do it in a way that it is not a repetitive we could compute a large number of partial derivatives at one go.

So, before we look at the mathematical details we just get an intuition for backpropagation.

(Refer Slide Time: 02:11)

We need to answer two questions

- How to choose the loss function $\mathcal{L}(\theta)$?
- How to compute $\nabla\theta$ which is composed of $\nabla W_1, \nabla W_2, \dots, \nabla W_{L-1} \in \mathbb{R}^{n \times n}, \nabla W_L \in \mathbb{R}^{n \times k}, \nabla b_1, \nabla b_2, \dots, \nabla b_{L-1} \in \mathbb{R}^n$ and $\nabla b_L \in \mathbb{R}^k$?

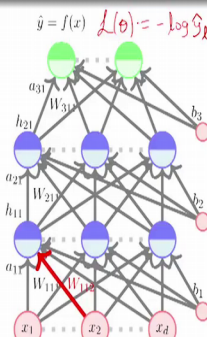


Mitesh M. Khapra CS7015 (Deep Learning): Lecture 4

And then we will get into the gory details of how to actually compute these gradients and partial derivatives. So, this is the portion that we are in we are intended to ask these two questions. And this is where we are.

(Refer Slide Time: 02:21)

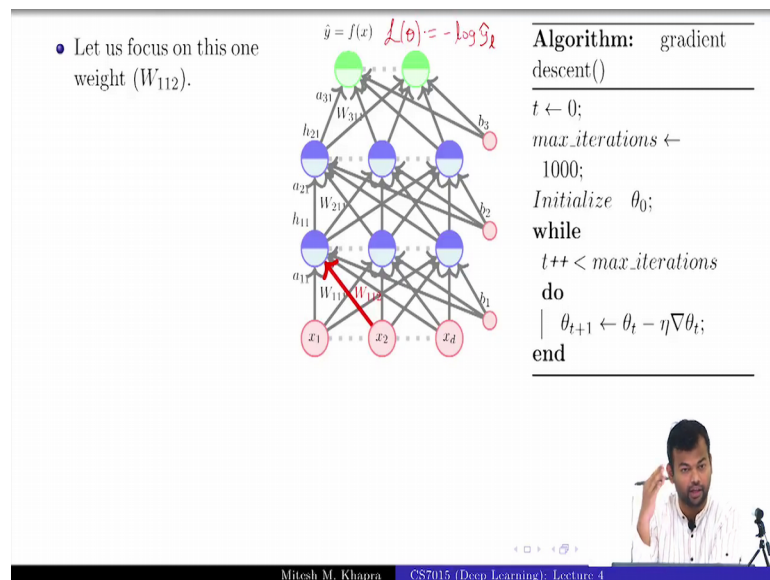
- Let us focus on this one weight (W_{112}).



$\hat{y} = f(x)$ $\mathcal{L}(\theta) = -\log \sigma_2$

Algorithm: gradient descent()

```
t ← 0;
max_iterations ← 1000;
Initialize  $\theta_0$ ;
while
  t++ < max_iterations
do
  |  $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \theta_t$ ;
end
```



Mitesh M. Khapra CS7015 (Deep Learning): Lecture 4

So now this is what our network looks like, this is clearly much more complex than that single neuron that we had, and which had only 2 weights W and b that was very easy to compute the gradients there. Now imagine that I want to compute the gradient of the loss

function, right? And let us assume it is a classification problem then what is the loss function minus log of y hat.

So, this is the loss function, and we want to compute the derivative of this with respect to one of these weights in the network. And am deliberately taking something which is much farther away from the loss, but why do you say why do I say it is much farther away, it is right at the input layer, right? And the loss is somewhere at the output layer. So, we want to compute this gradient.

(Refer Slide Time: 03:10)

- Let us focus on this one weight (W_{112}).
- To learn this weight using SGD we need a formula for $\frac{\partial \mathcal{L}(\theta)}{\partial W_{112}}$.
- We will see how to calculate this.

Algorithm: gradient descent()

```

t ← 0;
max_iterations ← 1000;
Initialize  $\theta_0$ ;
while
  t++ < max_iterations
  do
     $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \theta_t$ ;
  end

```

Mitesh M. Khapra CS7015 (Deep Learning): Lecture 4

Now, to learn sorry you want to learn this way, to learn this weight we know that we can use gradient descent. We are all convinced that this gradient descent algorithm which I have shown here, as long as we put all these variables or all these parameters that we have into theta.

We can just run the gradient descent algorithm and compute, then the only thing that we will need is this partial derivative with respect to all the weights in the network. And in particular with respect to this weight that I am interested at, ok.

(Refer Slide Time: 03:40)

- First let us take the simple case when we have a deep but thin network.
- In this case it is easy to find the derivative by chain rule.

$$\frac{\partial \mathcal{L}(\theta)}{\partial W_{L11}} = \frac{\partial \mathcal{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{L11}} \frac{\partial a_{L11}}{\partial h_{21}} \frac{\partial h_{21}}{\partial a_{21}} \frac{\partial a_{21}}{\partial h_{11}} \frac{\partial h_{11}}{\partial a_{11}} \frac{\partial a_{11}}{\partial W_{L11}}$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial W_{L11}} = \frac{\partial \mathcal{L}(\theta)}{\partial \hat{y}} \frac{\partial h_{11}}{\partial W_{L11}} \quad (\text{just compressing the chain rule})$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial W_{211}} = \frac{\partial \mathcal{L}(\theta)}{\partial h_{21}} \frac{\partial h_{21}}{\partial W_{211}}$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial W_{L11}} = \frac{\partial \mathcal{L}(\theta)}{\partial a_{L1}} \frac{\partial a_{L1}}{\partial W_{L11}}$$

26/57

Mitesh M. Khapra
CS7015 (Deep Learning): Lecture 4

Now so, we will now see how to calculate this, we will first this is only to get the intuition, right. So, we will first think of a very simple network, which is a very deep, but the thin network, ok, it has many layers, but it is a very thin network, here you see what I mean by a thin network, ok. Now this is what I am interested in. Can you tell me how to compute this? This looks like a chain. So, it is justified the user chain rule of derivatives, right. So, what would the chain rule look like?

You want to compute the derivative of this with respect to this. And you have done this in high school, right. So, you have functions of the form of sine of cos of tan of e raised to sine of x, right and this is exactly how this chain is right, you have some function of x followed by another function of x, another function of x function of x function of x and so on. You just keep making a composite function of the input. We actually wrote down that function if you remember. It was just one function applied after the other function, right or a very composite function. So, you just need to apply the same idea here. Does this look ok? So, we take we go step by step. So, I am almost accounting for every shade of color here.

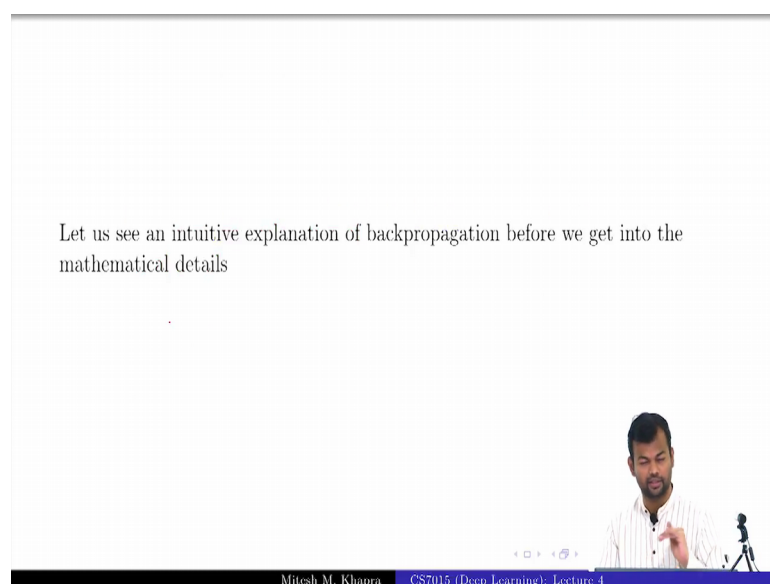
So, $\frac{d\theta}{dy}$, then $\frac{dy}{da_{L11}}$, there is only one neuron here, then this with respect to the sorry h_{21} , then h_{21} with respect to a_{21} , a_{21} with respect to h_{11} , h_{11} with respect to a_{11} , and then a_{11} with respect to W_{111} .

So, I just traversed down the chain in the reverse order, this is how the chain rule works, right? Anyone has a problem with this? It is straight forward? And now what I have done is for convenience I have just compressed the chain. You see the red part and the green part, I have just compressed this weight. So, that and this is again something that you have done in high school if you have this you could just write the chain as the first and the last it. So, this is what you can do, and I also compress this other chain, and am going to use these kinds of compressions later on.

So, what am trying to impress on you is that, if I want to go from here to here, right that is what my intention is. If somehow I have already travels from here to here, then I can just reuse that computation? That is the idea which I am trying to impress on it. I do not need to follow the entire chain every time, I can do these partial computations up to a point, have you seen this something similar idea somewhere else? Dynamic program is something like that. So, you have just computed up to a certain point, and then it is reuse the value for further down the chain.

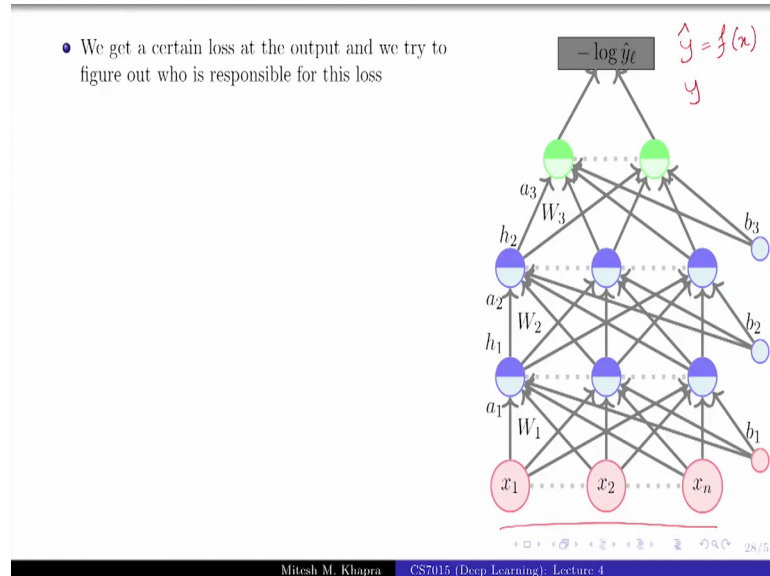
So, that is what we are going to do. And same for all the weights, right. For each weight the chain size would be different depending on where it lies in the network, for the weights which are very close to the output layer the chain would be very small, makes sense? So, this is the intuition and we will see the intuition a bit more.

(Refer Slide Time: 06:40)



So, let us now understand this in the terms of the wide complex network that we are using..

(Refer Slide Time: 06:47)



So, what actually is happening is that, we are at a certain stage; that means, we have some values of W's and b's ok, at the initial stage we just have these W knots and b knots. But let us assume that we have done some training, and we are at a certain level we are at W_t at time step t and b_t at time step t , for all the weights inverse. Now we feed it a training example, we do this entire compute computation, what do we get at the end? We get \hat{y} which is a function of this x that we have fed it. But we also know this true y , right? We know the true value we know \hat{y} .

So, we can compute the loss function. So, we compute the loss and to our surprise we see that the loss is not 0, we are getting a non 0 loss; that means, the network has not yet learnt properly, if the weights and biases are still not in the right configuration that we want them to be in right. So now, what do we do? We go on this path of investigation, we want to find at who in this network is messing up things? There is someone who is causing this problem, because of which I am not getting the desired output. And we are on our quest is to now find out who this guy is who is responsible for this. So, what would you do? Where would you start? The output layer.

(Refer Slide Time: 08:11)

- We get a certain loss at the output and we try to figure out who is responsible for this loss
- So, we talk to the output layer and say “Hey! You are not producing the desired output, better take responsibility”.
- The output layer says “Well, I take responsibility for my part but please understand that I am only as good as the hidden layer and weights below me”. After all ...

$$f(x) = \hat{y} = O(W_L h_{L-1} + b_L)$$

28/57

Mitesh M. Khapra CS7015 (Deep Learning): Lecture 4

Because the output layer is the guy who give you the output, right. So, go and talk to him, and we say that hey what is wrong with you why are you not producing the desired output, right? Now what is the output layer going to tell you? In very civil language, I will say I cannot do anything boss, I mean, I was just given some weights and inputs from the previous layer, and those weights and inputs were messed up.

So, there is nothing which I can do go and talk to them. So, who will it directors do? It will say that I am just as good as $W_L h_{L-1} + b_L$, and be it because these are the guys that I completely depend on. If these guys were ok, then I would have been fine. So, we then go and talk to these guys, that what is wrong with you.

(Refer Slide Time: 08:58)

- So, we talk to W_L, b_L and h_L and ask them “What is wrong with you?”
- W_L and b_L take full responsibility but h_L says “Well, please understand that I am only as good as the pre-activation layer”
- The pre-activation layer in turn says that I am only as good as the hidden layer and weights below me.
- We continue in this manner and realize that the responsibility lies with all the weights and biases (i.e. all the parameters of the model)
- But instead of talking to them directly, it is easier to talk to them through the hidden layers and output layers (and this is exactly what the chain rule allows us to do)

$$\frac{\partial \mathcal{L}(\theta)}{\partial W_{111}} = \frac{\partial \mathcal{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3} \frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2} \frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1} \frac{\partial a_1}{\partial W_{111}}$$

Talk to the weight directly
Talk to the output layer
Talk to the previous hidden layer
Talk to the previous hidden layer
Talk to the weights

29/57

Mitesh M. Khapra CS7015 (Deep Learning): Lecture 4

So now they say fine W_n and b_L take the responsibility, they are the nice guys, they say we are the weights, we are supposed to make a , we are the ones who are responsible for the adjustments in the network. So, we have failed to do our job properly, and I think we should get adjusted right. But then h_L will resist, it will say it is not my fault, why will it resist? Because it again depends on the previous activation layer.

So, till then point as to what? The W 's and b 's in the previous layer, right. And you see how the investigation is now proceeding where will we reach? Well keep going down the network, we are talking to everyone in the network, we are talking to every dark green guy every light green guy, every dark blue guy every light blue guy, we are also talking to all these weights and biases. And in the end what do we figure out? The responsibility lies with all the weights and all the biases. They are the ones who are responsible for this.

Now, ok, but now we find out that this is also one of those weights which is responsible, and this is also one of these weights that is responsible. But it was have been very difficult for us to talk to them directly. So, then what are we going to do? Instead of talking to them directly which is this, we will talk to them through the chain rule. So, we will talk to the output layer that is exactly how what we did maybe went to the first guy that we knew, that guy pointed out to the previous hidden layer, that guy pointed us to the previous hidden layer. And then finally, we get to the weights.

So, this talking to is fine, but where do derivatives figure out in this, why are why is the language derivatives, why are we not talking in English or Hindi or something else? What does the derivative tell us? So, talking about gradient descent like what we saw in gradient descent, but in general what does the derivative tell us? If I change this a bit, how much does my loss change.

So, that is how much this guy is responsible for the loss, because if this is very sensitive, even adjusting a bit of this I could drastically reduce the loss. So, that is what the derivative tells us, that tells us how sensitive is the loss function to the weight or any quantity with this with respect to which am taking the derivative. That is why the language is of derivatives? Is that clear? Is the intuition fine to everyone?

(Refer Slide Time: 11:29)

Quantities of interest (roadmap for the remaining part):

- Gradient w.r.t. output units
- Gradient w.r.t. hidden units
- Gradient w.r.t. weights and biases

$$\frac{\partial \mathcal{L}(\theta)}{\partial W_{111}} = \underbrace{\frac{\partial \mathcal{L}(\theta)}{\partial y}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial y}{\partial a_3}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_3}{\partial h_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial h_2}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$

- Our focus is on *Cross entropy loss* and *Softmax* output.

So now will convert this intuition into actual math and try to figure out. How to compute every guy along the way, right? And we will use this idea that we have made some partial computations and then well, use it for the rest of the chain. So, we have made this much, at some point, we will reach where we have made this much, and then you could use it for the rest of the chain. In fact, we will start right from here well start with this guy and then keep expanding the chain.

So, the rest of the story is going to be about computing 3 quantities. Can you tell me which are these 3 quantities? Gradients with respect to the output units, gradients with respect to the hidden units, and then gradients with respect to the parameters, ok. So,

these are the 3 things that we need to do, if we do this we have everything in the chain, and we are done, ok? And the other thing that we need to do is, we cannot sit down and compute this for every element, right, we want to have it in a generic fashion, where instead of talking about W_{111} , W_{112} and so on.

We should at least be able to talk about W_1 , W_2 and so on. So, that means, we have only 3 matrices and 3 biases, right at least at that level. So, we have to do a collective computation instead of just computing for every guy. So, instead of looking at scalars, which is what we are doing when we are doing gradient descent for W and v ? We were just computing the update rule for W and b , we want to now do it for vectors and matrices.

So, that is that is the transition that is going to happen. And our focus is going to be on, what cross entropy and softmax why is that important, because that is the loss function. So, that is the quantity that am going to take the derivative, if I change the loss function all the gradients are going to change. Are all the gradients going to change? Only the first guy will change in the change, all this should remain still same, right? Modulus some conditions, but largely it should remain the same right, unless you change something in between. So, is the set up cleared everyone.

So, that is the end of this module.