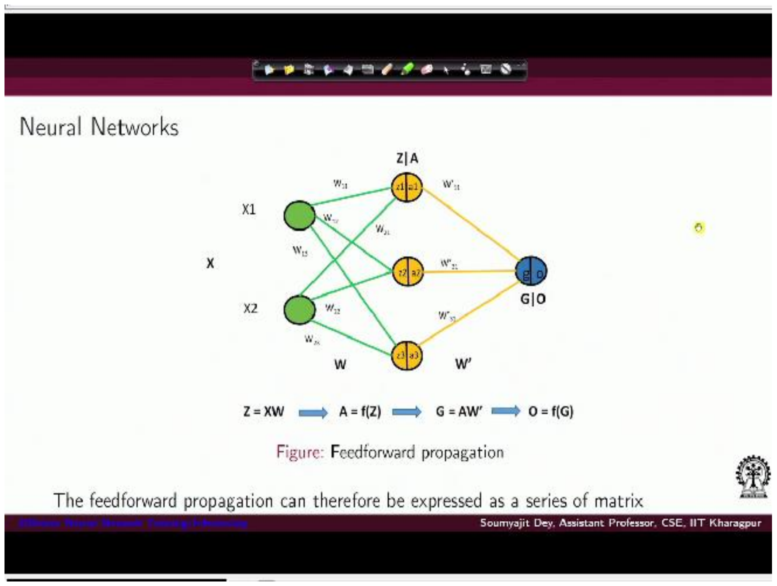
GPU Architectures and Programming

Prof. Soumyajit Dey
Department of Computer Science and Engineering
Indian Institute of Science Education and Research-Kharagpur

Lecture-56
Efficient Neural Network Training/Inferencing (Contd.)

Hi, welcome back to the lecture series on GPU architectures and programming. So, in the previous lecture, we just took an example of a simple neural network.

(Refer Slide Time: 00:30)



Neural Networks

Figure: Feedforward propagation

The feedforward propagation can therefore be expressed as a series of matrix

Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

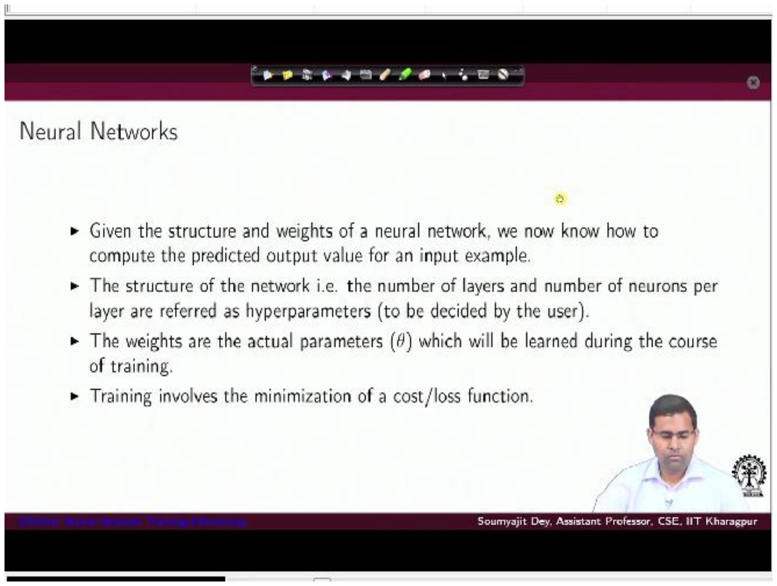
And we are trying to show how the inputs flow through these linear maps and then nonlinear activation functions and finally reach the output layer. Next, we will try to set up the loss function for such a neural network and see that how it I mean how the weights are being trained in a typical neural network. So, these are the basic architecture that we have. As you can see that if we start now using vector notation.

And try to understand what is the mathematical relations that are holding up here. So let us say this is my input X. And if I consider a matrix representation of all these weights let that be W. So, what we have here, reaching the hidden layer is some Z, which is nothing but X multiplied by W right. So you have X multiplied by W and that is the Z which is reaching this and then if denote using f, the activations that are happening here.

Then after the activation with they get output matrix A, which is f of Z f represents the activation function. So we have A that is the activation flowing through this and then this activation captured by this vector A gets multiplied by this, again, a set of linear maps denoted by these W primes. So let us capture it by the capital matrix W prime. So you have A W prime giving us the input matrix flowing into this which is some G.

And then on this G we again have the activation function f applying again by f we mean a set of activation functions applying component wise and that gives me the final output right. So, we have I mean if we just summarize we have some input X with a linear transformation W followed by f and then again the linear transformation W prime and then again we have the f that gives me the output right.

(Refer Slide Time: 02:35)



Neural Networks

- Given the structure and weights of a neural network, we now know how to compute the predicted output value for an input example.
- The structure of the network i.e. the number of layers and number of neurons per layer are referred as hyperparameters (to be decided by the user).
- The weights are the actual parameters (θ) which will be learned during the course of training.
- Training involves the minimization of a cost/loss function.

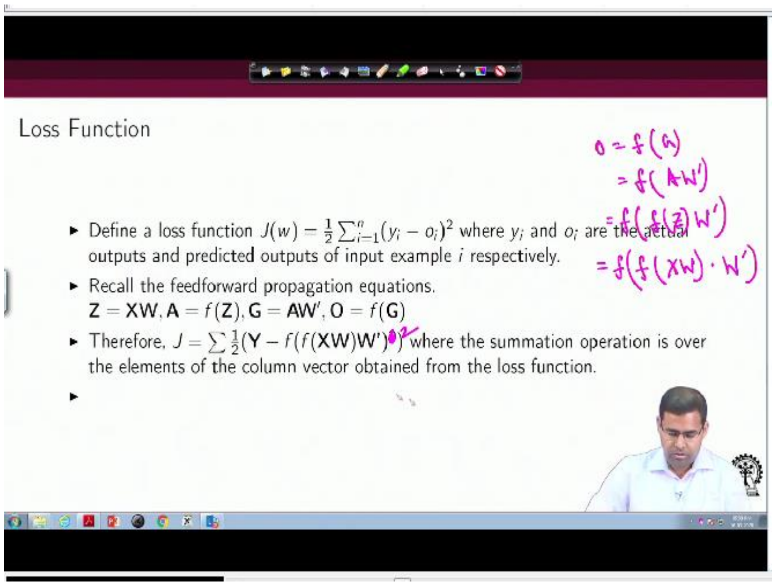
Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

So, what do we really want to do here. So, we are given the structure of the neural network, we are trying to find out what I mean we are trying and suppose we are given the weights here, we now know how to compute the predicted output from the input example, right. Now, of course, the overall objective is I am given the structure here and I am trying to figure out what are the suitable weight value.

So that the underlying overall function, which as we can see is basically a sequence of compositions here functional compositions. How does it best approximate the overall input output relationship right. So that is the training job. Now, the structure of the network, there is a number of layers, the number of neurons per layer and all these. These entire architecture, that is what we call we denote by mathematically as the hyper parameter, right.

And the weights represent the actual parameters, which you want to learn here, right. So, essentially, we will do the same thing like we discussed for the simple perceptron example of binary linear classifier that will set up a loss function for this kind of a vector matrix system. And we will see that how on that loss function we can perform suitable tuning of weight parameters and minimize the loss function.

(Refer Slide Time: 03:59)



Loss Function

- Define a loss function $J(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - o_i)^2$ where y_i and o_i are the actual outputs and predicted outputs of input example i respectively.
- Recall the feedforward propagation equations. $\mathbf{Z} = \mathbf{XW}, \mathbf{A} = f(\mathbf{Z}), \mathbf{G} = \mathbf{AW'}, \mathbf{O} = f(\mathbf{G})$
- Therefore, $J = \sum \frac{1}{2}(\mathbf{Y} - f(f(\mathbf{XW})\mathbf{W'}))^2$ where the summation operation is over the elements of the column vector obtained from the loss function.
-

$$\begin{aligned} O &= f(G) \\ &= f(AW') \\ &= f(f(Z)W') \\ &= f(f(XW)\cdot W') \end{aligned}$$

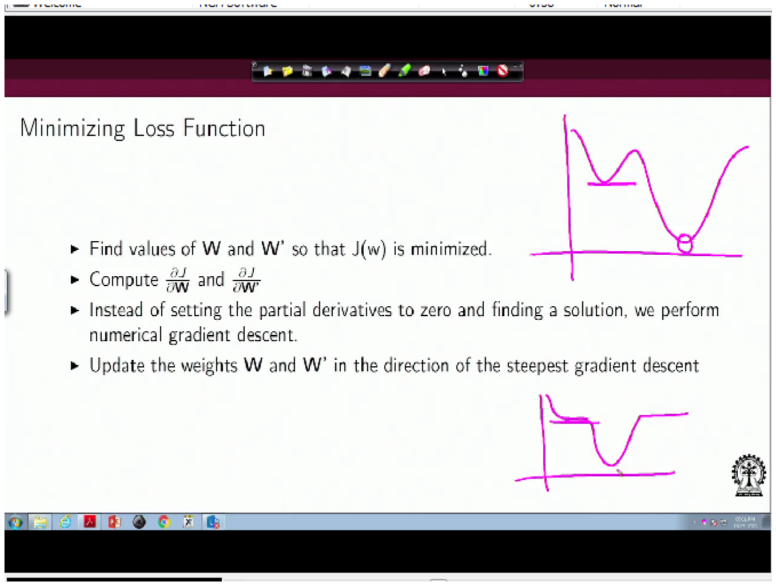
So, we consider a simple mean square loss function. So, that is essentially you consider the overall errors, you have a test data set where you have the actual y's and these o's are the computed outputs right. So, this is your mean squared error loss function. So, y i;s and oi’s are the actual outputs and predicted outputs for a given input example x i like that. So, you do the square sum it up and I mean considering n number of values.

So, you will have this mean squared error like this. And then let us understand how does this get represented by the overall matrix system that we set up earlier. So, as we can see, output f output is like a function of, so if we write it down here f of G. So that is f of AW prime. So that is f of f

Z, W prime. So that is f of f of XW, and then W prime, right. And so if I use the vector notations to read the loss function, we have something like this, right.

So J is we will have the summation outside and then y minus, y is the output. And this is what the function relationship gives me f of XW, multiplied by W prime square, W prime and finally at the output, I have this square, so that would mean so, this is what is W prime and then this there right. So, the square is about right.

(Refer Slide Time: 06:15)



Minimizing Loss Function

- Find values of W and W' so that J(w) is minimized.
- Compute $\frac{\partial J}{\partial W}$ and $\frac{\partial J}{\partial W'}$
- Instead of setting the partial derivatives to zero and finding a solution, we perform numerical gradient descent.
- Update the weights W and W' in the direction of the steepest gradient descent

So, overall what is the requirement, these W and W primes are my unknowns and I want this loss function to be minimized right. Now, as we know the way to do a function minimization is that you do a derivative calculation with respect to the variable which is W and W prime and figure out what are the minimized and I mean put the values back and then you get the minimum value for the function right.

But observe that here we have this W and W prime as vectors right. So, we need to compute the partial differentials which are del J del W and as well as del J and del W prime, right. And also we have several problems here right because even if I compute, what are the points where this partial derivative set to 0, and we find a solution, we need not be sure that those are exactly the only minima.

Because for a complex function, for example, let us take a simple example here, this may be a complex function right you can have a local minima and being a global minima. So, you consider one solution you may get you can get stuck in this kind of a local minima. And you can miss the global minima, which one is this, or there can be several options, maybe you can get stuck at this kind of foot hill, right, which is kind of flat.

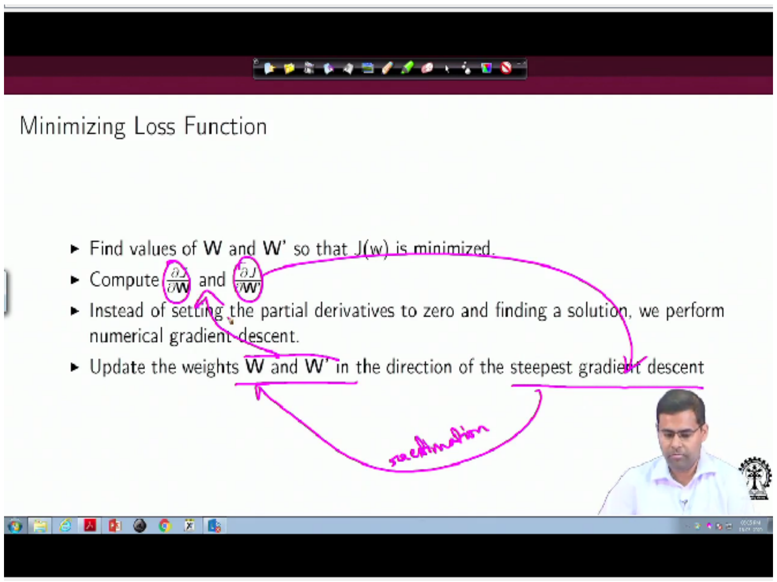
And you think that you have got the solution without sampling the other territories and you do not really get here, right. So, overall, it does not really make sense for a sufficiently complex system where these W's and W primes are significantly big, I mean, it may not really make sense for such complex functional relationships that you really try to set the partial derivative to 0 and get a minima.

Rather, it makes sense to actually do some optimization here in the way that you perform what we call as numerical gradient descent. That means, you try to figure it out at different points, you compute this del J, del W and del J del W prime, and you try to figure out what are the directions in which these values, the value of the loss function decreases with the highest possible gradient, right.

Because then that is the maximum slope. So you should really focus your search on those reductions. I hope I am making myself clear, let me just repeat again, since J will be in a sort of sufficiently complex situation, this J has I mean J is, I mean, these W's and W primes of a very high dimension, right and J will have a very, very complex functional form. So, rather than really trying to set the partial derivatives to 0.

And compute all such minimas, which may be quite intricate, it makes sense I mean, using exact closed form expressions, it makes sense that you try to evaluate, you first find out I mean, how does these things look like right.

(Refer Slide Time: 09:31)



Minimizing Loss Function

- Find values of W and W' so that J(w) is minimized.
- Compute $\frac{\partial J}{\partial \mathbf{W}}$ and $\frac{\partial J}{\partial \mathbf{W'}}$
- Instead of setting the partial derivatives to zero and finding a solution, we perform numerical gradient descent.
- Update the weights **W** and **W'** in the direction of the steepest gradient descent

So, you will get some expressions, then, on those expressions, if you figure out what are the value of those expressions for your current estimates of both W and W prime. Then you get that well, for my given value of W and W prime at those values, the rate of change of the loss function is something like this. From those let us of change, you get that well, in which what is the direction in which the loss function is decreasing maximally right.

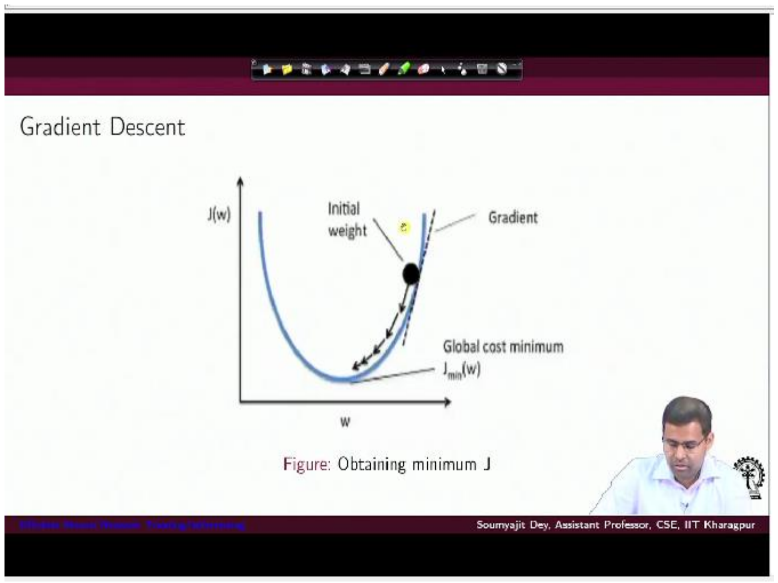
And then you start taking a different estimate of the value of the loss function what you do is when you are trying to figure out what are the directions in the space of W and W prime on which the directions in which the loss function is minimizing maximally. So that is giving you the steepest gradient descent and then in these directions, you do a reestimate of both W and W prime. And accordingly, you again re-evaluate these values right.

So, you use to find I mean use these parameter values to figure out what is the direction of the steepest gradient descent in that direction, you do a re-estimation. After that estimation, for those modified values of W and W prime, you again compute these values. You carry on doing this until or unless you get convergence or you see that they are not really descending right. Now, again, just to let you know and of course, most of most of you are aware of this already that for sufficiently complex systems, even this is going to be difficult to do.

And what people really do is very smart algorithm called stochastic gradient descent right. We are not going to that detail for the time being rather, we will just see that what are the underlying computations, which are involved for finding out these values, because as you can see, whatever will be your heuristic for computing the gradient descent and finally, figuring out what should be the values of W and W prime I am going to be happy with every time you need to compute this del J, del W and del J del W primes.

So, what we are really interested in is what is the mathematical as well as the computational perspective of complexity of these values right.

(Refer Slide Time: 11:54)


Gradient Descent
J(w)
Initial weight
Gradient
Global cost minimum
$J_{min}(w)$
w
Figure: Obtaining minimum J
Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

(Refer Slide Time: 11:58)

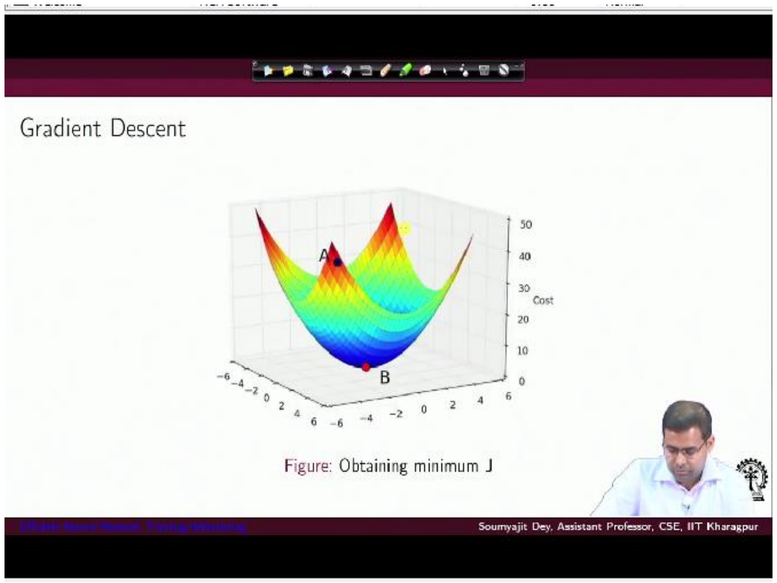
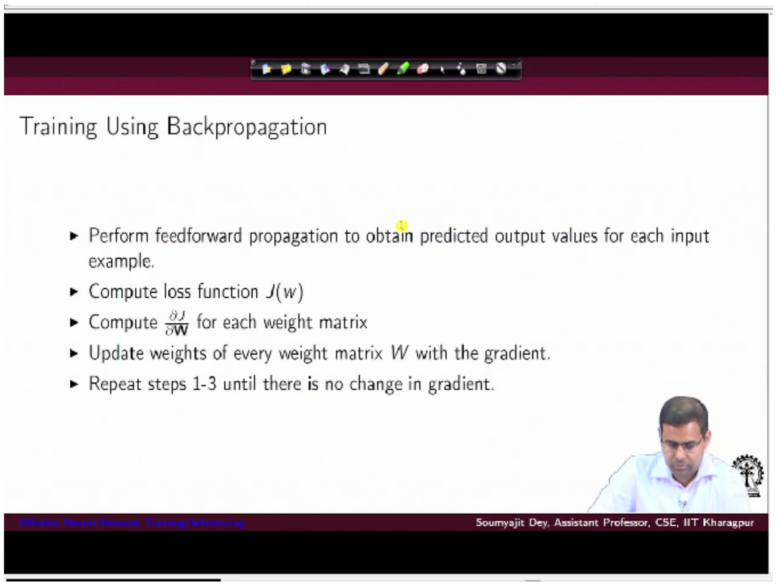


Figure: Obtaining minimum J

Yeah.

(Refer Slide Time: 12:00)



Training Using Backpropagation

- Perform feedforward propagation to obtain predicted output values for each input example.
- Compute loss function $J(w)$
- Compute $\frac{\partial J}{\partial W}$ for each weight matrix
- Update weights of every weight matrix W with the gradient.
- Repeat steps 1-3 until there is no change in gradient.

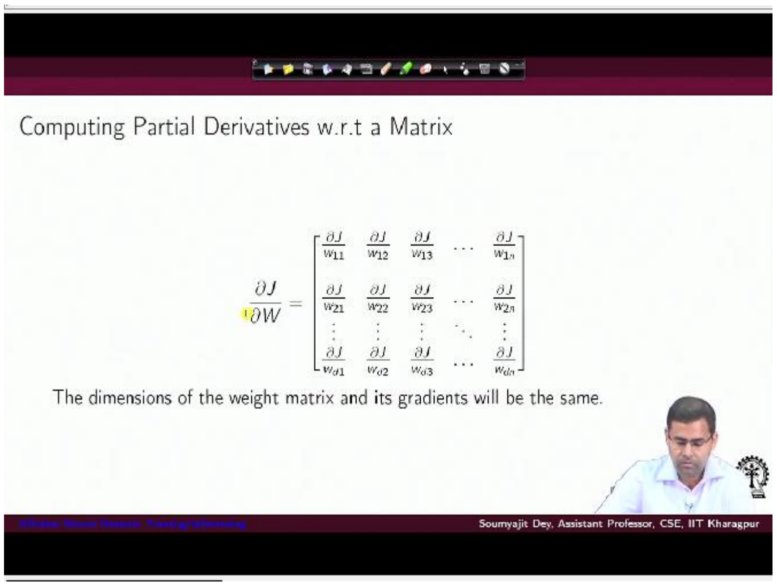
Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

So that is essentially what we call as training using backpropagation. So, the first step is you perform feedforward propagation to obtain the predicted values, you obtain the predicted values using those predicted values, you compute the loss function, then you compute the value of, of course, let us understand what is the relation, given the X, you need to compute the Y. That is the forward pass. Why do you really want to do that.

Unless you get the Y's that your network estimates you cannot really figure it out what is the error that means what is the difference between your estimate and the actual values that are there for the outputs in that data set, right. That is why you need the feedforward propagation only then you can do the loss function value estimation. And then you compute what is the value of del J del W for the current weight values, right.

Once you have computed this, you tried for each of the weight matrix, you have to do the reestimate, right. So that is the face of updating the weights for every weight matrix with the gradients, right. Once that is done for this modified values, you again do the entire thing. That means computing the loss function and all that.

(Refer Slide Time: 13:16)



Computing Partial Derivatives w.r.t a Matrix

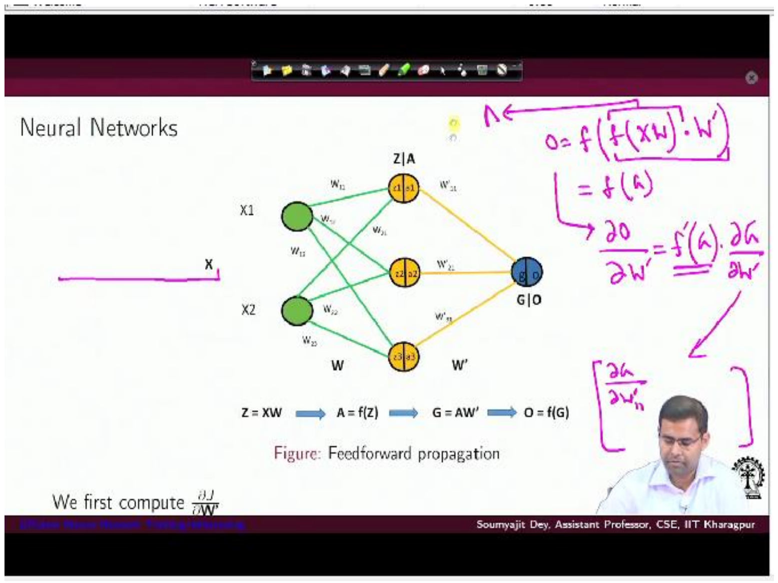
$$\frac{\partial J}{\partial W} = \begin{bmatrix} \frac{\partial J}{w_{11}} & \frac{\partial J}{w_{12}} & \frac{\partial J}{w_{13}} & \cdots & \frac{\partial J}{w_{1n}} \\ \frac{\partial J}{w_{21}} & \frac{\partial J}{w_{22}} & \frac{\partial J}{w_{23}} & \cdots & \frac{\partial J}{w_{2n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J}{w_{d1}} & \frac{\partial J}{w_{d2}} & \frac{\partial J}{w_{d3}} & \cdots & \frac{\partial J}{w_{dn}} \end{bmatrix}$$

The dimensions of the weight matrix and its gradients will be the same.

Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

So let us see, what is the underlying matrix operations that are going on. So what is del J del W, when you express this in terms of components, as we have seen for the capital W, it is essentially having all these components right. So let us go back to our nice small neural network to see the components of W.

(Refer Slide Time: 13:34)


Neural Networks

$Z = XW \rightarrow A = f(Z) \rightarrow G = AW' \rightarrow O = f(G)$

Figure: Feedforward propagation

We first compute $\frac{\partial J}{\partial W'}$

$O = f(\underbrace{f(XW)\cdot W'}_{G})$

$= f(G)$

$\frac{\partial O}{\partial W'} = f'(G)\cdot\frac{\partial G}{\partial W'}$

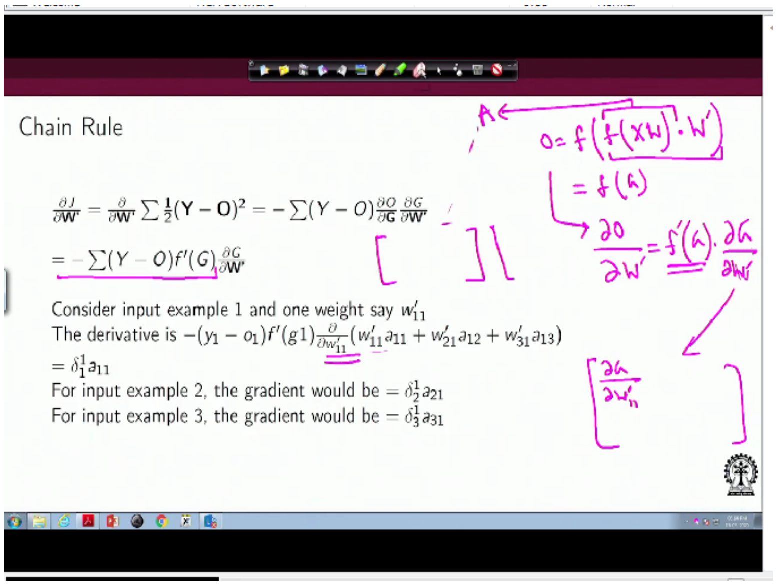
$\left[\frac{\partial G}{\partial W'_{11}} \quad \right]$

Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

So, these were my components for a simple 3 cross 2 system right. So, you have this W 11 W 12 W 13, again W 2 and W 2 W 23. So in that way you have like this. So these are the W 1 components that are W 2 components like these W d components, right like that, right. So with this we will first see what is going on in the feedforward propagation and that is what we have. So, as we have seen in the feedforward propagation, what we get was this output, output is nothing but activation inside this f again on the input followed by W multiplied by W prime.

And then again the activations use my feedforward propagation right. Now, using output I need to figure out what is my J and then do and then we have to do the computation of del J del W prime right.

(Refer Slide Time: 14:36)



Chain Rule

$$\frac{\partial J}{\partial \mathbf{W'}} = \frac{\partial}{\partial \mathbf{W'}} \sum \frac{1}{2}(\mathbf{Y} - \mathbf{O})^2 = -\sum (Y - O)\frac{\partial O}{\partial \mathbf{G}}\frac{\partial G}{\partial \mathbf{W'}}$$

$$= \underline{-\sum (Y - O) f'(G)}\frac{\partial G}{\partial \mathbf{W'}}$$

Consider input example 1 and one weight say w'_{11}

The derivative is $-(y_1 - o_1)f'(g1)\frac{\partial}{\partial w'_{11}}(w'_{11}a_{11} + w'_{21}a_{12} + w'_{31}a_{13})$

$= \delta_1^1 a_{11}$

For input example 2, the gradient would be $= \delta_2^1 a_{21}$

For input example 3, the gradient would be $= \delta_3^1 a_{31}$

$$A \leftarrow \quad O = f(\boxed{f(XW)' \cdot W'})$$

$$= f(G)$$

$$\frac{\partial O}{\partial W'} = f'(G) \cdot \frac{\partial G}{\partial W'}$$

$$\begin{bmatrix} \frac{\partial G}{\partial w'_{11}} & \\ & \end{bmatrix}$$

So, what is this del J del W prime is nothing but summation like this. So, of course, what we get is del J del W prime summation 1 by 2 and we have this Y minus O square. Essentially this is my mean square function right where that we are talking about. And then of course, what is going to happen is you are taking the derivative with respect to W prime okay first we are trying to figure out what is del J del W prime and then we will also figure out what is del J del W right.

So, here as you can see this will give us something like I mean this square term will come down and we will have the summation followed by the output minus I mean the estimates. The estimates and then you have this finally, we want the derivative with respect to W prime. So, you will now start applying this W prime on the O right because Y is that some actual value that you have observed right.

So, I mean this is the value that is really there in your test data, right. And this is the value that you have actually observed right. So, y minus O is something that you have as an output of the feedforward propagation. But the next thing is what you do this, you get the functional form because you are trying to do a differentiation of del O del W prime. So that gives you del since O is nothing but is basically del O I mean, if we just write it in terms of the intermediate steps. So O was like some f G right. Because we were calling this entire thing G in our expression. So accordingly we write it as del O del G del W prime, right.

So now as you can see, this is the form that we get. No, but we just represent since I am doing del O del W prime, that would naturally give me f prime G followed by del G del W prime. So what is f prime G is nothing but a derivative of the activation function. Now, that is easy to compute, because for the activation function, we are considering simple differentiable functional forms. So it is nothing but the differentiable some first derivative of that functional form, right.

So the trick really is part of the last component which is del G del W prime, and that is something that will prevail with. Now, for the big neural network or all the components there will be many terms. Let us take as an example, just the first term. So, for our input example, let us just considered the first weight which was W 11 prime right because as you can see, that this del G del W prime is going to have many components right.

This is going to have many components like just like del J del W, this del G del W prime will also have a lot of components, and we are just trying to see what should be the first component. So, considering W 11 prime, which would be one of the components. So, del G del W 11 prime that is what we are going to get right because well, what is G. So, as you can see G is nothing but some value multiplied by W prime right.

What is this value, f of XW. If you look back, this was nothing but my A right. Something that we actually computed here. Yeah, is so A is f of X W, right. And that is a value that we already have, which is components right. So then this reduces too. So I am just considering one of the components and it will reduce to W 11 prime A 11 plus W 21 prime A 12 plus W 31 prime A 13, so on so forth, right.

I mean, why are we really saying that let us elaborate it out. So, we are considering this del G del W prime and as we have seen W prime will have different components. Now, we are considering one component which is del G del W 11 prime and inside G what we have is this A which is nothing but f of XW right. And considering that we have already evaluated A Ei nd that A has got these values, right.

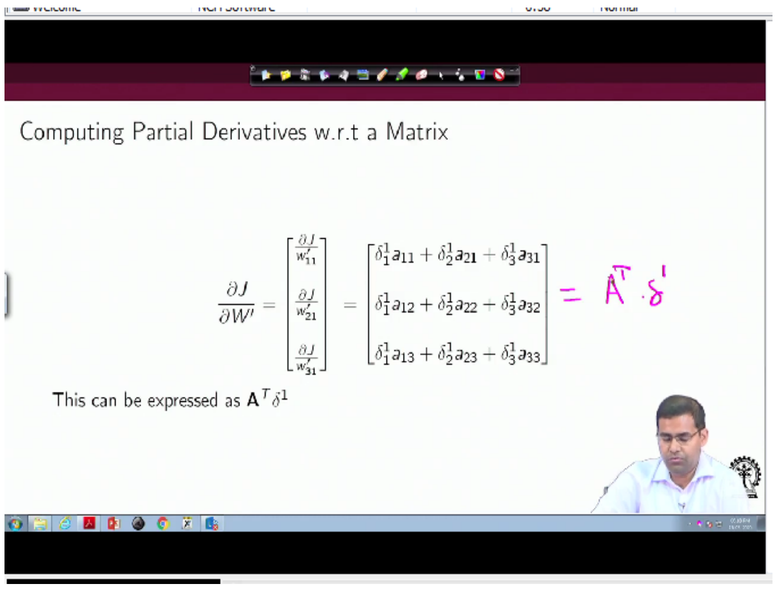
So, what are the points at which A is going to interact with W 11 prime. Well, that would be only the first point, right. So, just to make it more clear here, so how does A really look like because we are considering A followed by W prime right. So, A will have a matrix form and then we have W prime with all those values right. And then if we just take the first part, then I have W 11 prime multiplied by A 1 plus W 21 prime A 2 W 31 prime A 13 like that.

And as you can see if I am applying the partial differential with respect to W 11 prime only, then I will just get the first component of A, which is A 1 alright, sorry, I mean, we will just so, if we take the derivative and only consider first component, we get this kind of a value of A 1 1 and the previous part is same, right because that is not going to change. Now, similarly for the other inputs examples, if we consider the other weights right.

Then similarly, we will be able to figure out the other components where some scalar multiplied by A 21 some scalar multiplied by A 31 and all that right. Now, as you can see that all these A's are something we already have available as part of the feedforward computation, but the part that is coming out here is this delta 1 matrix right. Because as we can see for del 1, we have these components delta 11 delta 21 delta 31 like I mean delta 11 delta 12 delta 13 like this, where this delta 1 is nothing but this part.

So, just like for y 1 O 1 and f G 1 we have delta 1 first component. Similarly we can compute the other components.

(Refer Slide Time: 21:58)



Computing Partial Derivatives w.r.t a Matrix

$$\frac{\partial J}{\partial W'} = \begin{bmatrix} \frac{\partial J}{w'_{11}} \\ \frac{\partial J}{w'_{21}} \\ \frac{\partial J}{w'_{31}} \end{bmatrix} = \begin{bmatrix} \delta^1_1 a_{11} + \delta^1_2 a_{21} + \delta^1_3 a_{31} \\ \delta^1_1 a_{12} + \delta^1_2 a_{22} + \delta^1_3 a_{32} \\ \delta^1_1 a_{13} + \delta^1_2 a_{23} + \delta^1_3 a_{33} \end{bmatrix} = A^T \cdot \delta^1$$

This can be expressed as $\mathbf{A}^T\delta^1$

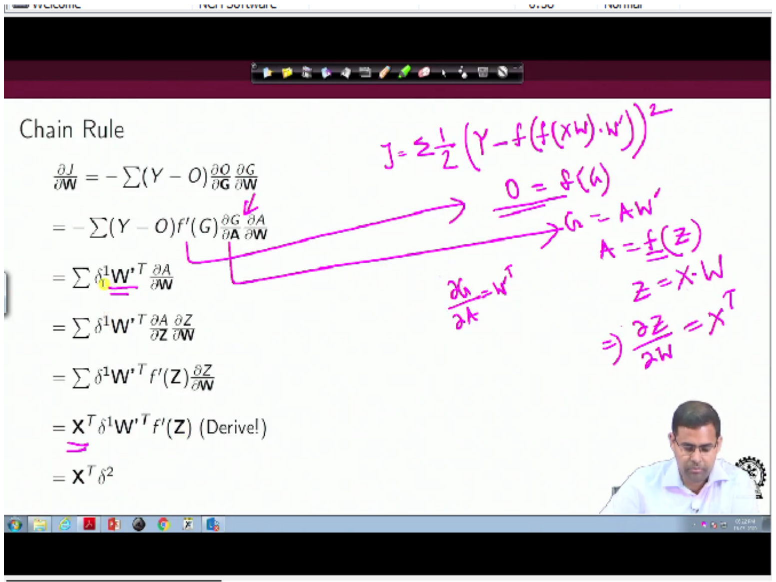
And if we consider like this, then overall representation that we will get will be something like this right. Just like we have the delta J delta W prime. From that we have figured out what is the functional form considering only the way to W 11 prime and the example one. If we keep on considering the other examples, overall, the functional form that we will have for del J del W prime would be something like this right.

Because w prime is going to have these different components like W 11 prime 12 1 prime, 31 prime like that. And of course, there will be other elements here. And for each of them, we have to figure out this right. I mean, this delta 11 a 1 delta 2 1 and delta 12, a 21 delta 13 a 31. Like this, right. Yeah. So, I hope this is clear, because as we can see for w prime, these are my components, right w 11 prime W 21 prime, W 31 prime.

And all we are doing is for the overall partial derivative, we are finding each of these terms component wise. And as you can see for each of these terms, if we consider each weight, I mean each input example separately, we get the component terms for the derivatives, right. And overall, this is what we arriver right. Now, one way to write it would be because as you can see that we have the entire a matrix here in a transpose form, right.

So, let us say it is just nothing but A transpose delta 1. I hope it is clear why we are writing a transpose A delta 1 because here as you can see, for the different components, we are essentially getting A in the transpose for now, just like we computed del J del W prime, we can do a similar thing for del J del W, right.

(Refer Slide Time: 24:23)



Chain Rule

$$\frac{\partial J}{\partial \mathbf{W}} = -\sum (Y - O) \frac{\partial O}{\partial \mathbf{G}} \frac{\partial G}{\partial \mathbf{W}}$$

$$= -\sum (Y - O) f'(G) \frac{\partial G}{\partial \mathbf{A}} \frac{\partial A}{\partial \mathbf{W}}$$

$$= \sum \delta^1 \mathbf{W'}^T \frac{\partial A}{\partial \mathbf{W}}$$

$$= \sum \delta^1 \mathbf{W'}^T \frac{\partial A}{\partial \mathbf{Z}} \frac{\partial Z}{\partial \mathbf{W}}$$

$$= \sum \delta^1 \mathbf{W'}^T f'(\mathbf{Z}) \frac{\partial Z}{\partial \mathbf{W}}$$

$$= \mathbf{X}^T \delta^1 \mathbf{W'}^T f'(\mathbf{Z}) \text{ (Derive!)}$$

$$= \mathbf{X}^T \delta^2$$

$$J = \sum \frac{1}{2} \left(Y - f(f(XW) \cdot W') \right)^2$$

$$O = f(G)$$

$$G = AW'$$

$$A = f(Z)$$

$$Z = X \cdot W$$

$$\Rightarrow \frac{\partial Z}{\partial W} = X^T$$

$$\frac{\partial G}{\partial A} = W'^T$$

So, what about del J del W. So, again, we will do a derivative right. So, this one square has come out right. So, minus summation Y minus O, now, you have del O del G del G del W right. So, if we just elaborate the left hand side here. So, essentially what you will get is J is summation half y minus I am just writing the O part in the expanded form right. So that is what we have here, right. And then when I take the derivative with respect to G.

So what do we get del O del G because I am taking the derivative of del O del W right. So, I will make it del O del G del G del W. So, again what we can do is we have del O del G and we have O is nothing but f of G. G is nothing but AW prime as we have seen, and this A is nothing but f of Z and Z is as we have seen the original input, multiplied by the weight of the first layer, right. So that is the setting we have.

So as you can see, when you del O del G, that is nothing but is going to give you the f prime which is essentially the derivative of the activation function of the hidden layer. And then you have to perform this del G del W, which you have broken down into 2 parts. So you have to do del G del A, that means this expression, so what is really that going to give you. So, if you do I mean just by using normal linear algebra relation, I have G equal to AW prime.

If I take a derivative, del G del A, that is going to give me W 2 prime transpose and that is what we get. So we have the derivative here. Since G is AW prime. So del G I am sorry, del G del A is going to give me W prime transpose using standard linear algebra rules. So that is what we get and then I have del A del W. So now I will break this del A del W further to del A del Z and del Z del W. So what is del A del Z.

As you can see, A is equal to F Z again, I have an activation function and del A del Z is going to give me the differentiation of that activation function right. So that is again f prime of Z right. And then I have del Z del W right. So what is that. So again, for Z what do we have is X dot W. So naturally again, using standard linear algebra rules for product of matrices. If I do del Z del W, we are going to get is X transpose, right.

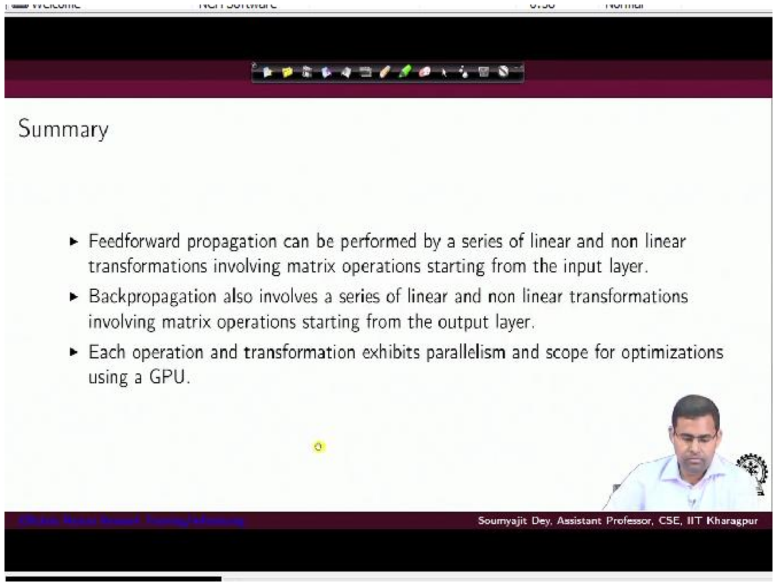
So that is what we have the X transpose. But the question is where did this so, we have solved for this part, how the W 1 W prime transpose comes, how the X transpose comes, how the f prime Z comes. But what about delta 1. Well, if you look back into our previous derivation, what was my delta 1. If you look here, delta 1 is minus summation, this thing right, y minus O times the f prime of G, right.

So just apply it here. So in that way, when you get this minus summation Y minus O f prime of G, that reduces to your delta 1, right. So you already have it here. And then when you do del Z del Z del W, that is going to reduce to your X transpose. And this is your final form. Here, you replace delta 1 with W prime transpose and f prime Z using a new constant which is delta 2, right.

So, we are just trying to figure out that how to derive this del J del W and this would be our expression fine. So, now, once we have figured out that will for del J del W, we have a representation in terms of A transpose del 1, right. That is my del JW prime sorry. And then for del J del W, we are every presentation, which is X transpose delta 2, we are going to use these values because they are giving me the gradients of J with respect to W.

And I will just use this gradient values to adjust the value of both W and W prime, right. And that is what happens with in case of back propagation. So, through after this forward pass, I have got output values, I have computed the loss function, I have applied the derivatives, I have figured out using these relationships that we have established from here, we have tried to figure out how you get del J del W prime and del J del W. And then you use those gradient fellows to adjust and compute to reestimate the values of W and W prime. And that is something that you know, back propagate.

(Refer Slide Time: 31:17)



Summary

- Feedforward propagation can be performed by a series of linear and non linear transformations involving matrix operations starting from the input layer.
- Backpropagation also involves a series of linear and non linear transformations involving matrix operations starting from the output layer.
- Each operation and transformation exhibits parallelism and scope for optimizations using a GPU.

Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

So as a summary, what we are really doing is I mean, we are re-computing the values of W and W prime and propagating that backwards through the neural network and again doing the forward pass that is computing the entire classifications output by using these new values of W and W prime, right. So, the way you will, I mean, if we if we just try to summarize the way the neural network training was working is.

You have a feedforward propagation which is performed by a series of linear and nonlinear transformations involving matrix operations starting from the input layer, and then you have a back propagation which involves again a series of linear and nonlinear transformations involving matrix operations starting from the output layer. What do we mean. See, we have gone through the forward propagation that gives me the value of delta 1, right.

You now have to apply delta 1 to this matrix to get what is delta 2 right. Because we have I mean, because you are going to use delta 12, I mean, to get a new value of this W prime, right. Because you have an original W prime, you have to subtract a transpose delta 1 first of all you compute delta 1 after doing the feedforward propagation, and then you readjust W prime with this relation, right.

Once and after that you also do what is like computing what is the delta 2, right because as you can see that computation of delta 2 clearly depends on delta 1. That is why we have a sequence here. After delta 1 is done, you can reestimate W prime using delta 1, you compute delta 2. Once you have computed delta 2, you reestimate the w values, right and set all those values again in the neural network. So that is your back propagation.

You are propagating back the values, readjusting values of W prime and W. So that is again a series of linear and nonlinear transformations but they start from the output layer. Now, as you can see that these operations actually have significant amount of computations like transpose, computation like matrix multiplication, things that we have already seen, parallel systems like GPUs are very smart at doing right. So that is why we will see that how GPU architectures can be exploited for doing all these computations. With this will end the current lecture. Thank you for your attention.