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

**Lecture - 04**
**Feed forward Neural Networks, Backpropagation**

So, welcome to lecture 4 of CS7015, the course on deep learning. Today, we will talk about feed forward neural networks and back propagation. So, quick recap of the story; so far it, so, we started with mp neurons. We saw there were some problems with the mp neurons. They could handle only Boolean inputs and Boolean outputs and if I show needed we hard coded. So, from there, we moved on to perceptrons which allowed for real inputs real outputs and sorry real inputs and binary outputs. And we also learned an algorithm for learning these weights and parameters right. So, we need there was no need to hand code these parameters anymore.

But then, we found that, for a single perceptron, there is a limitation. It cannot; it can only deal with functions which are linearly separable. So, then we went on to a multi-layer network of perceptrons and we proved by illustration that, it can handle any arbitrary Boolean function; whether linearly separable or not, the catch is that you will need a large number of neurons in the hidden layer, right. Then we also observed that perceptrons have this harsh thresholding logic. So, which makes the decisions very unnatural; it is 0.49, it is negative; 0.51 is positive; so, you wanted something more smooth.

So, the smoothest approximation to this step function which is the perceptron function was a sigmoid function. Sigma is a family of functions and we saw one such function which was logistic function. And then, we saw that, it is very smooth; now, it is continuous and differentiable.

Now, for the sigmoid neuron on a single sigma you know and we saw a learning algorithm which was gradient descent, And we proved principally that it will always go in the direction where the loss decreases right; so, that is what is the basis for gradient descent.

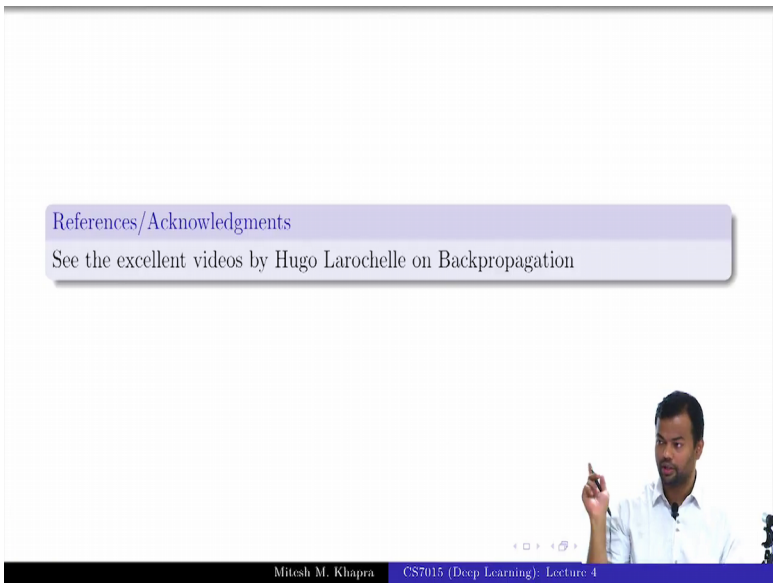
And then, we graduated from a single neuron to a network of neurons and made a case that such a network of neurons with enough neurons in the hidden layer can approximate any arbitrary function right, ok. So, I have told you that, it can approximate any arbitrary function. What does that mean? And what is the thing in the network that does all this? All the tower functions and the tower functions depend on weights and biases. So, there, in that illustrative proof, again we were adjusting the weights and biases by hand right? We knew that we wanted these very tiny tower functions and we were doing it.

Now, from there, where should we go?

Student: (Refer Time: 02:39).

We need an algorithm to learn these weights and biases right. So, that is what back propagation is. So, today I am going to formalize these feed forward neural networks. We just did it by illustration the other day. I will introduce you to the terminology and see what the input outputs are and so on. And then, we will look at an algorithm for learning the weights in this feed forward neural network, ok.

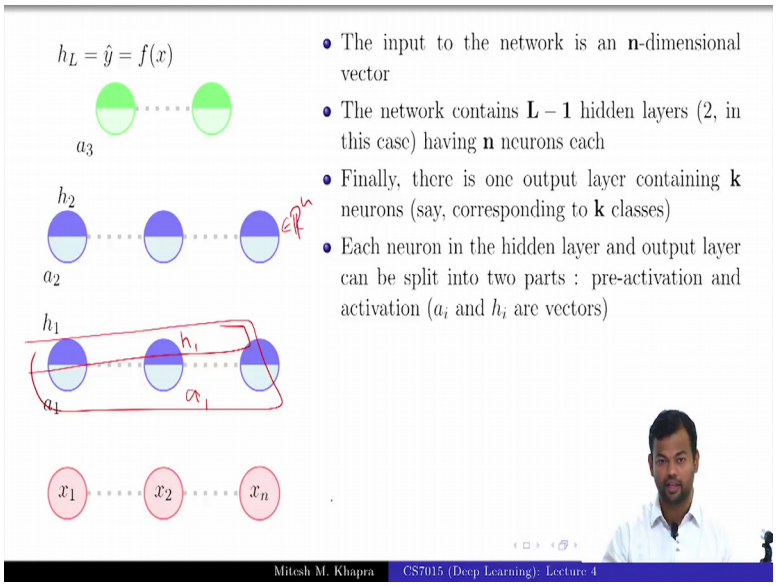
(Refer Slide Time: 03:03)




Let us begin. So this, a lot of this material is inspired by the video lectures by Hugo Larochelle on back propagation. He has a course on neural networks.

It is available on YouTube; you can check it, ok. So, let us first begin by introducing feed forward neural network, right.

(Refer Slide Time: 03:18)




So, what is a feed forward neural network? The input to the network is an n-dimensional vector so; that means, my input belongs to R n, but fine. The network contains L minus 1 hidden layers. Where do you already know what hidden layers are right? We have been defining that terminology since multi layered perceptron. So, you have these hidden layers and there are L minus 1 of these and then it has one output layer containing k neurons, ok. Those are the feed forward neural network looks like. What is missing here?

Student: (Refer Time: 03:57).

The weights, right.

So, each neuron in the hidden layer, ok, before that each neuron in the hidden layer and the output layer can be split into 2 parts right. So, I will call the first part as the pre activation and the second part as the activation. Have you seen this plate before right? What does the pre activation do?

Student: Aggregation.

Aggregation and what does the activation do?

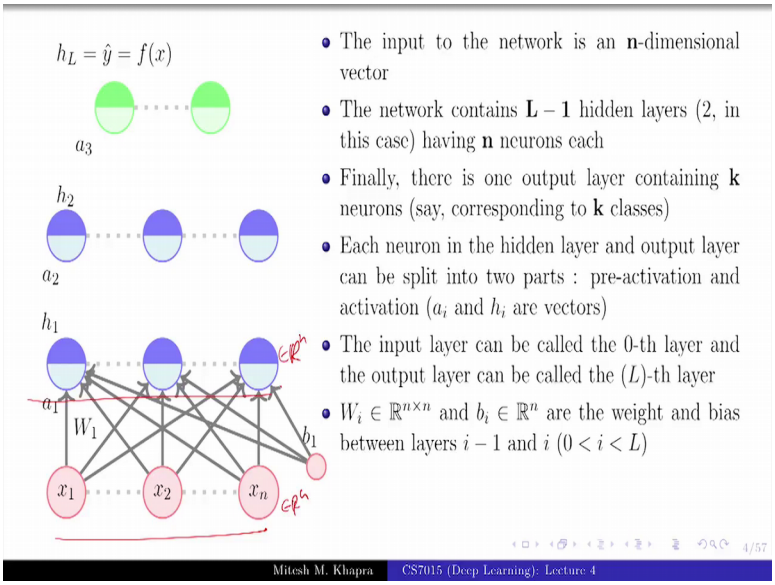
Student: Non-linearity.

Non-linearity, right? So, we have this pre activation and activation at every layer and a i and h i are vectors. Is that correct? Because, this entire thing or rather this part is h 1 and

this part is a 1. Both of these are vectors, right. And for this discussion, am going to assume that, everything till here belongs to R n, ok.

So, the input was R n and all the hidden layers also have n neutrons. Is that fine? So, please pay a lot of attention to this couple of slides because, this is going to stay with us for the rest of the lecture and perhaps 2 more lectures and even for the course alright. So, this is very important that you understand this; the way we are defining a feed forward neuron network.

(Refer Slide Time: 05:08)

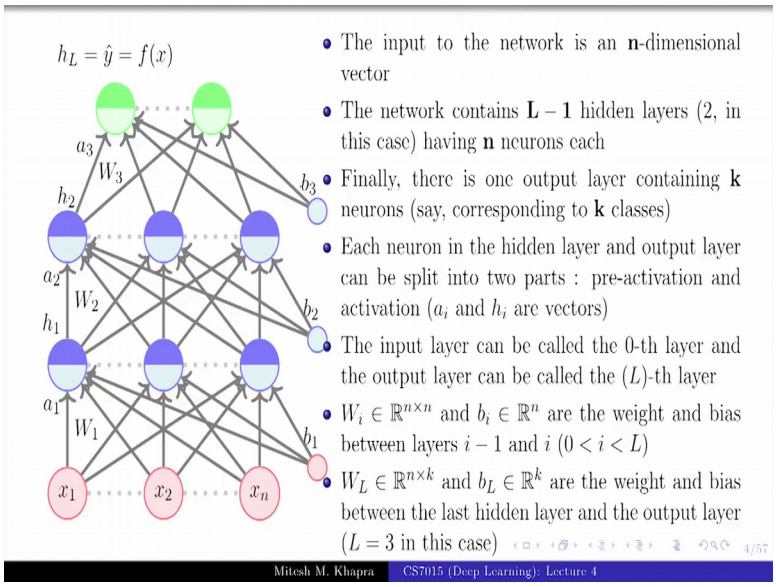



The input layer can be called as 0th layer. What I mean by that is that, I could refer to this as h 0, ok. There is no a 0 h 0 here because, there is no pre activation, activation. You are just given the input. So, I just call it as h 0 ok. And the last layer can be called as h of L, right? Whatever you get from this green part, you will call it as h of L. Ok what is the dimension of h of L? R raised to k. It belongs to R k, because I have said here that, you have k neurons, each corresponding to k classes, ok.

Now, we have weights between the input layer and the first hidden layer. Now, can you tell me this belongs to R n. This also belongs to R n. So, what is the dimension of W 1? N cross n, right? Because it contains weights for connecting each of these inputs to each of these hidden layers; there are n here n there right. So, it is n cross n.

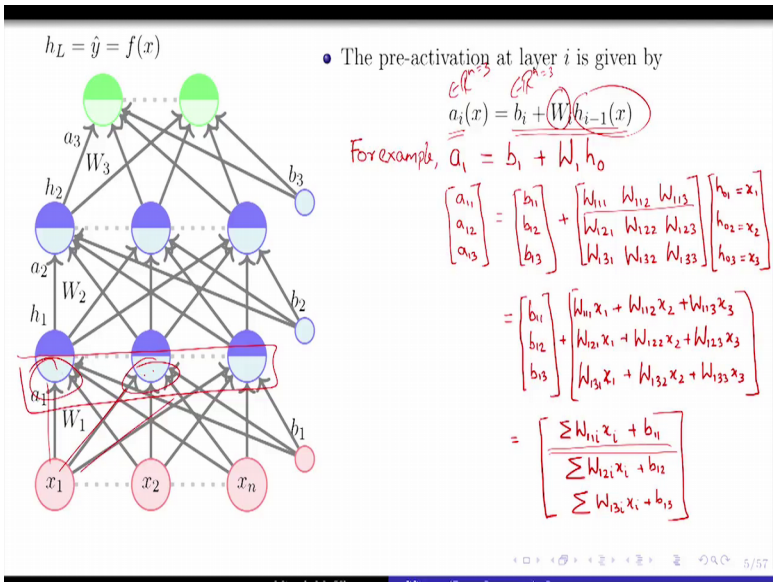
And what are the dimensions of the bias? N. One corresponding to each of the hidden inputs, fine. And this is only for up to this layer because, till here I have assumed everything is n.

(Refer Slide Time: 06:23)




Now, what about the output layer? n cross k and the biases k, k dimensional ok. So, this is what the network looks like. But now, I have to give you some function. So, I have just I have shown you a diagram, but what does it mean mathematically? Because, remember that, we are always interested in writing something of the form y is equal to function of x, right. And that is not well defined yet, ok.

(Refer Slide Time: 06:49)




So, let us start defining that. Ignore the red portion for now, ok. I will go over it. So, each of these activations right or rather the pre activations is given by b i plus W i into h i minus 1. So, what it means is that, these activations take inputs from the previous layer, multiply by them by weights and also add the bias. Is that clear? So, let us see it, right. For example, if I look at a 1 which is this vector. So, that is 3 dimensional and assuming it is 3 dimensional for simplicity.

So, it is a 1 1, a 1 1, a 1 2, a 1 3 right? And that is equal to how do you get rid off this? b 1 1 b 1 2 b 1 3 plus this matrix multiplication is this clear to everyone ok. I know it is trivial, but am still going over it right. So, let us not ok. And then, how do you do this matrix multiplication? Row was multiplied by the column i. So, this is what you get right? And in the end, I can write it as this, right? And this looks very similar to what we have been seeing throughout it from a mp neuron to perceptron to sigmoid neuron and now this case, right.

So, it is just an aggregation of all your inputs or weighted aggregation of all your inputs. That is the case which I want it to know; and that is obvious now. So, you understand what these are right.

So, this is R n. In our case, we have assumed n equal to 3. What is this? I will keep asking till this is completely fine with everyone. R n and this is?

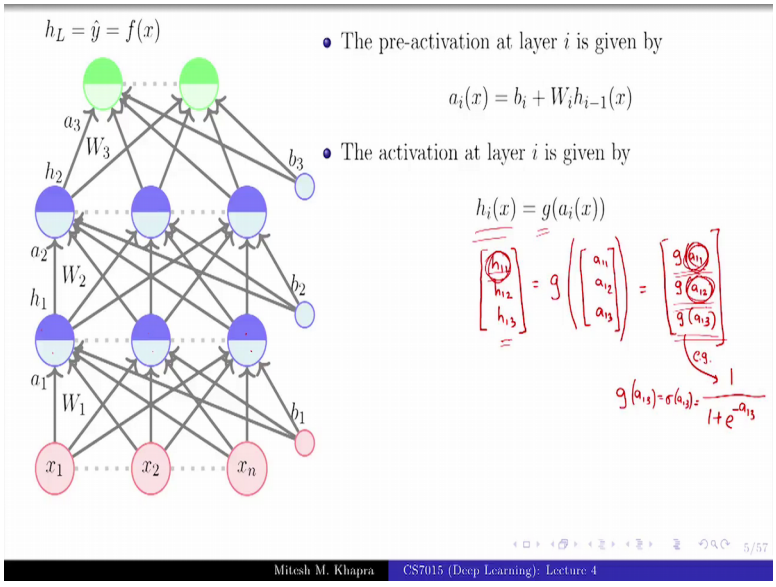
Student: (Refer Time: 08:30).

n cross n and this is?

Student: (Refer Time: 08:34).

n cross 1 n cross n I mean R n sorry is it fine? So, everyone understands the operation happening here. It is a weighted aggregation of your inputs. So, every guy here is a weighted aggregation of all the inputs, ok.

(Refer Slide Time: 08:47)

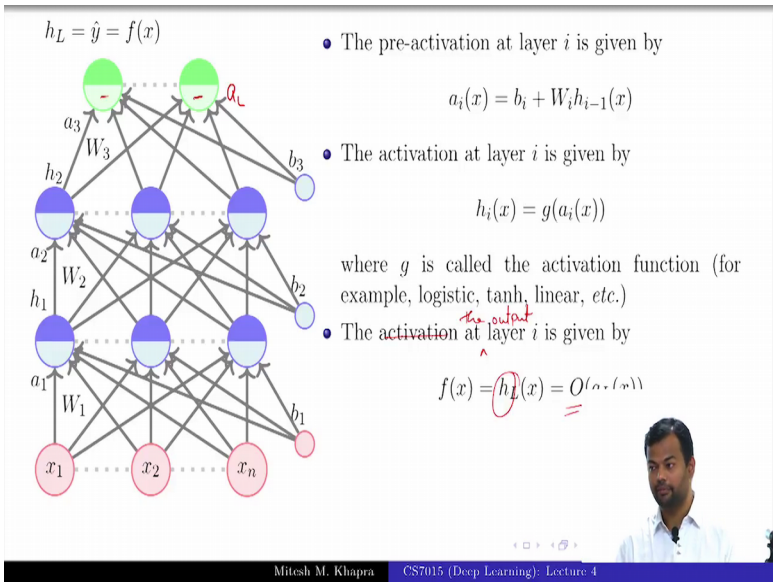



Now, after that I do h i of x is some function of a i of x, ok. What does this mean? So, this is again a vector, right? I have assumed that, it is 3 dimensional. So, these are the 3 elements of h i; so, these are the 3 guys. Now, these are some function of these light blue guys, ok. Now, how does that function operate on the vector? It operates element wise; not all functions on vectors are element wise. But this particular function, we are going to do element wise.

That means, that h 1 1 is equal to g of a 1 1, h 1 2 is equal to g of a 2 and h 1 3 is equal to g of a 1 3 right, where if I take g of a 1 3, one of the functions that I could choose is the sigmoid function. So, it would just be 1 over 1 plus e raised to minus here. So, what is happening is I am taking this value and passing it to the sigmoid function to get oh sorry am taking this value and passing it to the sigmoid function to get h 1 1 taking this value passing it to the sigmoid function to get h 1 2 right.

So, the key thing to understand here that, this is a element wise operation, right? It is not operating on the vector. That does not make sense. It is operating on every element of the vector, right ok. And g is called the activation function.

(Refer Slide Time: 10:02)

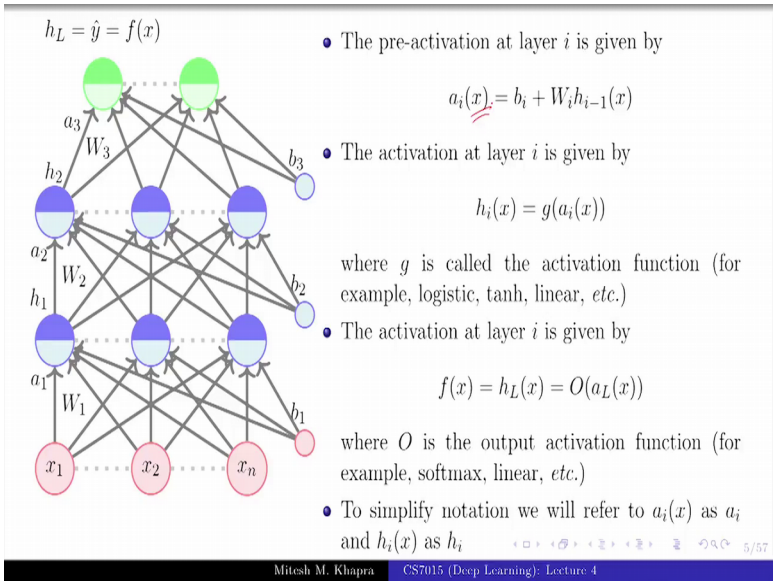



It could be logistic, tan h, linear anything right. So, we will see some of these functions later on ok.

Now, the activation at layer, I sorry they are supposed to be activation at the output layer..The activation at the output layer is given by the final function which is f of x is equal to O of a of. So, let us see; so, this is a 3. In our case, L was equal to 3 because, we had L minus 1 hidden layers and the Lth layer was the output layer right; so, this is a L. So, this is what I have computed here; that light green part of the figure that you see right now, based on that, I want to produce an output, right.

So, that is, someone had asked me a question that why do we always choose sigmoid? Because, sigmoid will clamp the output to 0 to 1. What if I want to predict the amount of oil which will not be between 0 to 1, right? That is why, for the output, we will use a spatial function that will call the output function and later on, I will show you that it depends on the task at hand ok. So, it is going to change with the task that we are going to do, right. So, we are just going to say that, the final output which is h of L is equal to some function of the pre activation at that layer. Is this terminology clear to everyone? How is each function operating, is that clear to everyone? Ok.

(Refer Slide Time: 11:22)




And we will see some examples of the output activation function right. Now just for simplicity am going to remove the x's from the bucket right; so, instead of calling everything a i of x h i minus of x and so on. I will just call them a i h i and so and so, that just simplifies things, but we know that everything is a function of x. Because, x is the input and that passes through some functions and we get the final output, right. So, this is the notations that we are going to use. Is the dimension of everything that you see every variable that you see here completely, clear to everyone?

Dimension of a i, b i. W. h i x, everything is clear ok. And the output layer has a slightly different dimension than the other layers because, there we have k classes as opposed to n neurons everywhere else, ok, fine. Now, I need to put this in the paradigm that, we saw for supervised machine learning. What were the five components there? Data.

Student: Model.

Model.

Student: Parameters.

Parameters.

Student: Learning.

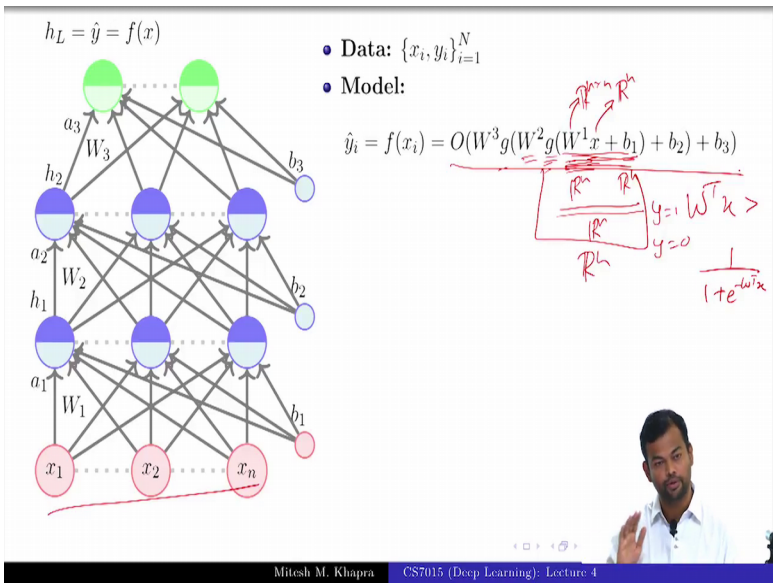
Learning algorithm.

Student: (Refer Time: 12:29).

Objective function right ok. Everyone remembers that ok?

So, I said that, we will do deep neural networks and we are trying to write this y hat as a function of x, but then, what I gave you is just a diagram from which this is not clear whether y hat is actually a function of x. How many of you think y hat is actually a function of x? Very few, ok.

(Refer Slide Time: 12:52)




So, let us see what exactly is our model assumption here, right. So, the question let me repeat the question just to be clear. So, I said that they are given some data we do not know the true relation between y and x we make an assumption that y is related to x using some function f right and it is has some parameters and then we like to try to learn the parameters of that function. So, what is the function here?

So, what is your model? What have you assumed as the model? Can you write y as a function of x? If yes, what is that function? How many of you have the answer? I think you have your answer, ok. I think I cannot wait more. So, I will give you the answer. Then it will become very obvious ok. So, this is how y is a function of x, right. So, let us see what is happening. I took the original x which was this, I transformed it, added b 1 that was the dash at layer 1.

Student: (Refer Time: 13:56).

No, this thing.

Student: (Refer Time: 14:00).

Reactivation at layer 1; I passed it through the activation function right, ok.

Now, again, let us be clear about the dimensions? What is the dimension of this?

Student: n.

n. What is the dimension of this? n cross n. So, what is the dimension of this product?

Student: (Refer Time: 14:21).

n. What about this? So, what is the product the final dimension of this?

R n. Now, you are passing it through a function g that function is operating element wise. So, what is the output dimension?

Student: R n.

R n. So, this is again R n, ok. Now this.

Student: (Refer Time: 14:42).

So, now you see the whole story, right. So, now, this n cross n guy multiplies with this n guy again. You get a vector again pass it through a non-linearity was it. So, high it is obvious now, right. You just take an x; just note down all the transformations that you have done. That is what a function does right. It passes it through the through first a linear transformation, this is a linear transformation, then a non-linear transformation, then again linear non-linear and so on, right.

So, just see how far we have come from where we started off, right? We started off with simple things like W transpose x, right? That was the perceptron model where we were taking decisions based on W transpose x and we were saying y is equal to 1. If this quantity is greater than something, y is equal to 0; if this quantity is greater than something right, that is why, we started off with we made it slightly more complicated by doing this. This was sigmoid neuron.

Now, from there, where have we gone to this right? So, we have increased the complexity of the network with great complicity. Complexity comes great?

Student: (Refer Time: 15:46).

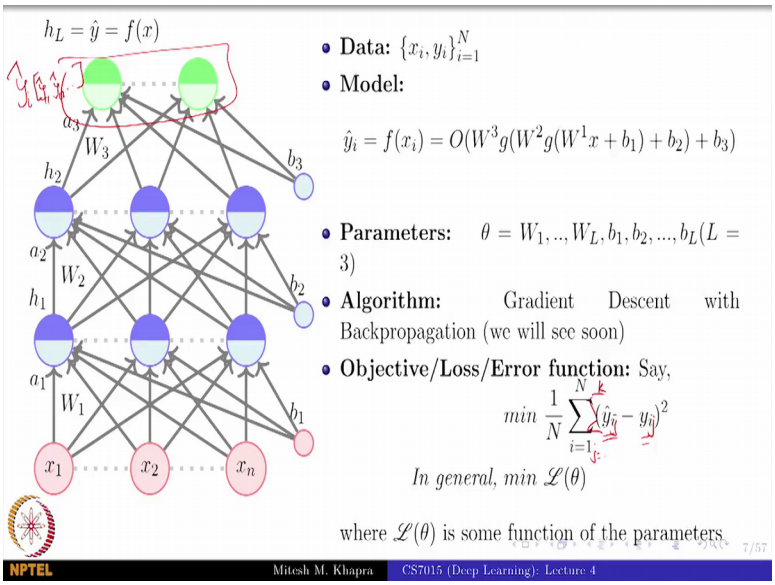
No power, right? We have already seen the representation power of deep neural networks, right. So, it comes from this complexity that you have you have a lot of linear and non-linear transformations, right? That adds to the complexity of the network. It has more parameters at each linear transformation you have some parameters and you are also using a lot of non-linearity. So, that is the reason why deep neural networks are so, powerful right do you get that? Ok, ok. So just to impress again, right.

So, any machine learning algorithm that you have you should be able to write it in this form right, that y is a function of x with some parameters and then your job boils down to learning these parameters, right. It just happens that here, y is a very complex function of the inputs. Is that clear? Ok. So, I am not deviated from the original story. I am still being able to write y as a function of x with some parameters, ok. What are the parameters?

Student: (Refer Time: 16:42).

All the W's, all the b's, right. So, W L to W n and b 1 to b L.

(Refer Slide Time: 16:44)



And the algorithm that we are going to see today for learning these parameters is called gradient descent, but we will use it with back propagation, where back propagation will help us to compute gradients. It is ok; it does not, it does not make sense at this point. That is what the lecture is supposed to be about, right. So, and what is an objective function?

Student: (Refer Time: 17:07).

Loss function. So, I could just go with this loss function, right ok. There is an error here. I thought we corrected this; there is a summation. So, actually these are vectors, right. So, this does not make sense. So, you should have summation j equal to 1 to k y i j minus y i j. Does that make sense? So, this is the vector y hat ok. For the i th example, it will be called as y hat high i which will have k elements, right. So, y hat i 1 y hat i 2 up to y hat i k, right.

So, that is what my predictions are and I will have the corresponding true vector also. I am trying to take the difference between them which is going to be an element wise difference. Everyone understands the error in the slide? How many of you do not get it? How many of you get it? If you do not get it, please raise your hands. It is a minor thing. I can correct it, ok. So, is the paradigm clear now? Ok. And how does deep neural networks fit into these this paradigm? Ok.