

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

Module - 3.5
Lecture - 03
Representation Power of a Multilayer Network of Sigmoid Neurons

So, before we move on to the next modulate some small corrections from yesterday's class.

(Refer Slide Time: 00:13)



For ease of notation, let $\Delta\theta = u$, then from Taylor series, we have,

$$\mathcal{L}(\theta + \eta u) \stackrel{\text{New}}{=} \mathcal{L}(\theta) \stackrel{\text{old}}{=} + \eta * u^T \nabla \mathcal{L}(\theta) + \frac{\eta^2}{2!} * u^T \nabla^2 \mathcal{L}(\theta) u + \frac{\eta^3}{3!} * \dots + \frac{\eta^4}{4!} * \dots$$

$$= \mathcal{L}(\theta) + \eta * u^T \nabla \mathcal{L}(\theta) \quad [\eta \text{ is typically small, so } \eta^2, \eta^3, \dots \rightarrow 0]$$

$$\nabla_{\omega, b} \begin{bmatrix} \frac{\partial \mathcal{L}(\omega, b)}{\partial \omega} \\ \frac{\partial \mathcal{L}(\omega, b)}{\partial b} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial^2 \mathcal{L}(\omega, b)}{\partial \omega^2} & \frac{\partial^2 \mathcal{L}(\omega, b)}{\partial b \partial \omega} \\ \frac{\partial^2 \mathcal{L}(\omega, b)}{\partial b \partial \omega} & \frac{\partial^2 \mathcal{L}(\omega, b)}{\partial b^2} \end{bmatrix}$$

NPTEL Mitesh M. Khapra Lecture 3

So, one was this partial derivative it should have been dou w square. So, we already taken one derivative with respect to w and now you are taking another derivative it is the gradient of the gradient, right. And similarly should this should have been dou b square, and this should have been dou w dou b, ok.

(Refer Slide Time: 00:43)

```
X = [0.5, 2.5]
Y = [0.2, 0.9]

def f(w,b,x) : #sigmoid with parameters w,b
    return 1.0 / (1.0 + np.exp(-(w*x + b)))

def error(w, b) :
    err = 0.0
    for x,y in zip(X,Y) :
        fx = f(w,b,x)
        err += 0.5 * (fx - y) ** 2
    return err

def grad_b(w,b,x,y) :
    fx = f(w,b,x)
    return (fx - y) * fx * (1 - fx)

def grad_w(w,b,x,y) :
    fx = f(w,b,x)
    return (fx - y) * fx * (1 - fx) * x

def do_gradient_descent() :
    w, b, eta, max_epochs = -2, -2, 1.0, 1000
    for i in range(max_epochs) :
        dw, db = 0, 0
        for x,y in zip(X, Y) :
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        w = w - eta * dw
        b = b - eta * db
```

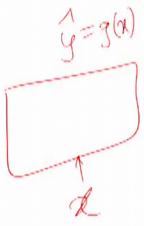
Gradient descent on the error surface

NPTEL Mitesh M. Khapra Lecture 3

The other small thing which I wanted to say was so, when I was executing this algorithm, right. So, I forgot to mention that just notice what is happening is the black dot that you see the black dots that you see, right and which are very close to each other. Actually, because you are just making small small movements those are the changes in the w comma b values and the red dots are the corresponding loss to that w comma b values right, just to clarify, ok.

So, that is why you see a movement on the w b plane which is this movement and as you keep changing that your loss function changes and it becomes better and better right; that means, it goes closer to 0.

(Refer Slide Time: 01:25)

<p>Representation power of a multilayer network of perceptrons</p> <p>A multilayer network of perceptrons with a single hidden layer can be used to represent any boolean function precisely (no errors)</p> 	<p>Representation power of a multilayer network of sigmoid neurons</p> <p>A multilayer network of neurons with a single hidden layer can be used to approximate any continuous function to any desired precision</p> <p>In other words, there is a guarantee that for any function $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$, we can always find a neural network (with 1 hidden layer containing enough neurons) whose output $g(x)$ satisfies $g(x) - f(x) < \epsilon$!!</p> <p style="text-align: center;">== \hat{y} y</p>
--	--

39/62

Mitesh M. Khapra Lecture 3

So, in this module we are going to talk about the representation power of a multilayer network of sigmoid neurons, right. So, I am going to compare these two things which are written in the title. So, first tell me; what was the representation power of a multilayer network of perceptrons. I roughly hear what you are saying and basically what you are telling me is that a multilayer network of perceptrons with a single hidden layer can be used to represent any Boolean function precisely, right. No errors that is; what we saw with that illustrative proof where we actually constructed once its network.

Now, what is the representation power of a multilayer network of sigmoid neurons? So, multilayer network of neurons with a single hidden layer can be used to approximate, ok. So, just see the difference in the language right. So, this was a represent; that means, exactly this is approximate; that means I will tolerate some error, any continuous function instead of Boolean function to any desired precision, right. So, this was not this was precisely with no errors this is up to any arbitrary desired precision.

So, what does this mean, actually what is the meaning of this? So, there is a guarantee that for any function, which takes our original x from \mathbb{R}^n to \mathbb{R}^m what is the m that we have been considering in all our examples one right we just care about one output, but it can be \mathbb{R}^m also. We can always find a neural network with one hidden layer containing enough neurons right. So, that is the operating trace here enough neurons whose output g of x , so, that means, you would have a network it would take as input $n \times 1$ it would

produce some \hat{y} and that is what I am calling as g of x , right, that g of x would be very close to the true function f of x , right.

So, remember that we said that there is this true function f of x which gives us the true y 's and we are trying to predict this \hat{y} . So, the true why I am calling by f of x and the \hat{y} I am calling by g of x and you can come up with a neural network which can give you values which can predict values which are very close to your true values, right. Does that make sense? Do you see the value of this theorem, what is it trying to tell me? Tell me, can you give me an interpretation of this? Why is this so useful? Do you know what this theorem is called universal approximations here and we did that in the history, right.

(Refer Slide Time: 03:57)

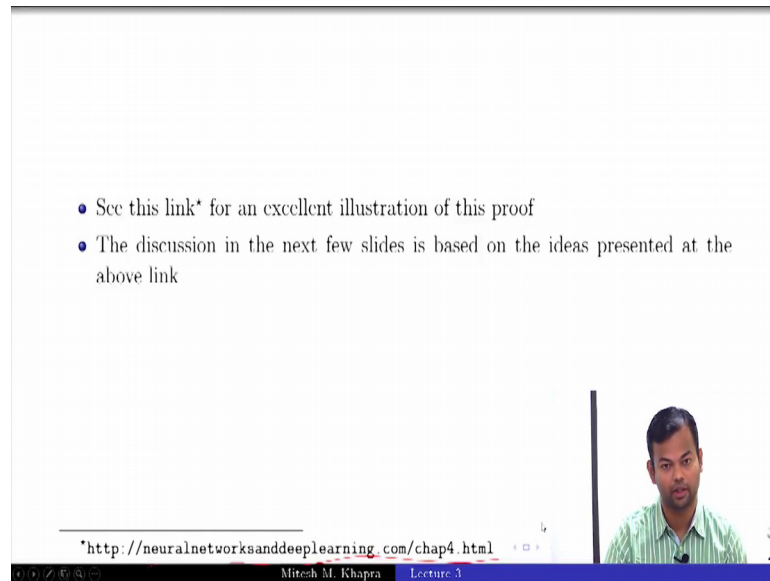
<p>Representation power of a multilayer network of perceptrons</p> <p>A multilayer network of perceptrons with a single hidden layer can be used to represent any boolean function precisely (no errors)</p>	<p>Representation power of a multilayer network of sigmoid neurons</p> <p>A multilayer network of neurons with a single hidden layer can be used to approximate any continuous function to any desired precision</p> <p>In other words, there is a guarantee that for any function $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$, we can always find a neural network (with 1 hidden layer containing enough neurons) whose output $g(x)$ satisfies $g(x) - f(x) < \epsilon$!!</p> <p>Proof: We will see an illustrative proof of this... [Cybenko, 1989], [Hornik, 1991]</p>
---	---

39/62

Mitesh M. Khapra Lecture 3

So, this was 1989, ok. What is the significance of this? Why do we care about such arbitrary functions and what does this theorem telling us actually? It is of course, telling us something about the representation power of a multi layer network of sigmoid neurons, but why is this important, ok. So, we will see that.

(Refer Slide Time: 04:26)



• See this link* for an excellent illustration of this proof

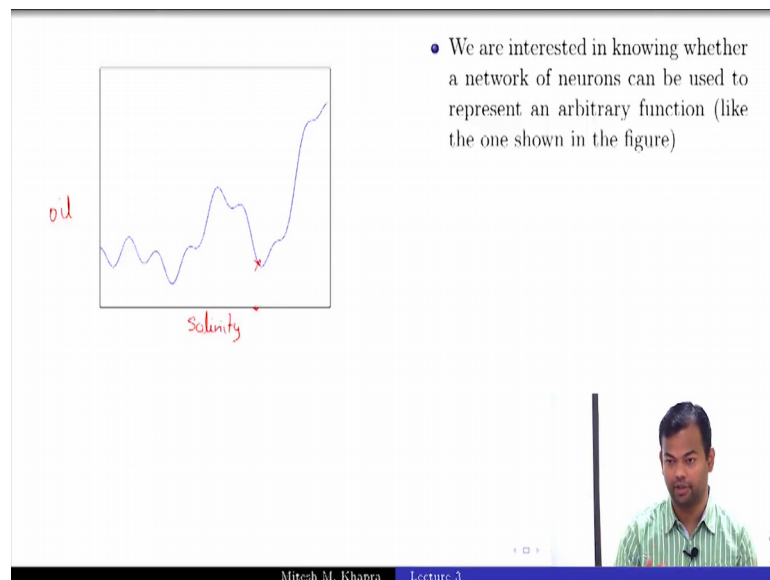
• The discussion in the next few slides is based on the ideas presented at the above link

<http://neuralnetworksanddeeplearning.com/chap4.html>

Mitesh M. Khapra Lecture 3

So, this the remainder of the lecture I have borrowed ideas from this URL you should actually read this it is a very interesting book it is available online for free very illustrative. So, please take a look at it, ok.

(Refer Slide Time: 04:41)



• We are interested in knowing whether a network of neurons can be used to represent an arbitrary function (like the one shown in the figure)

oil

Salinity

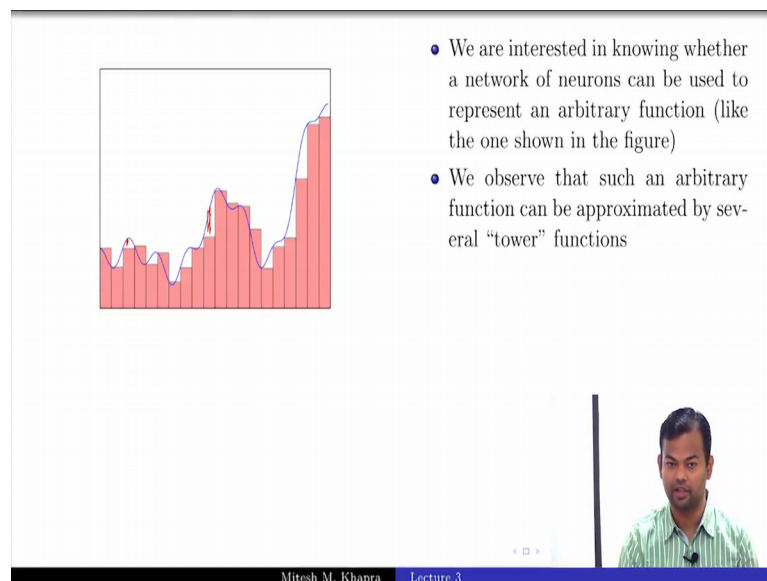
Mitesh M. Khapra Lecture 3

So, now actually what we are interested in is we are interested in knowing whether a network of neurons can be used to represent any arbitrary function like the one shown in the figure, ok. So, let me put some labels on this, so, they understand what I am trying to say. Suppose, this is salinity, again I go back to my oil mining example and I say that my

decisions are based only on a single variable which is salinity and this is actually how the amount of oil varies right as the salinity. It is a very arbitrary function, it is definitely not a linear function, it is not even a quadratic function, it is not an exponential function, it is just some arbitrary function, but a mathematical function this is possible. It is quite likely that salinities has this influence or in oil production or maybe it does not, but I am just taking that as an example, right.

Now, what do we want the network to learn? If I take some data and train the network at the end of training, what do I want? So, if I feed at this point after training, what should happen? It should give me this value, right that is what training means and that means, I should be able to approximate this curve, right, if I do that that means, I have learned from the training data, ok. So, let us see.

(Refer Slide Time: 06:00)

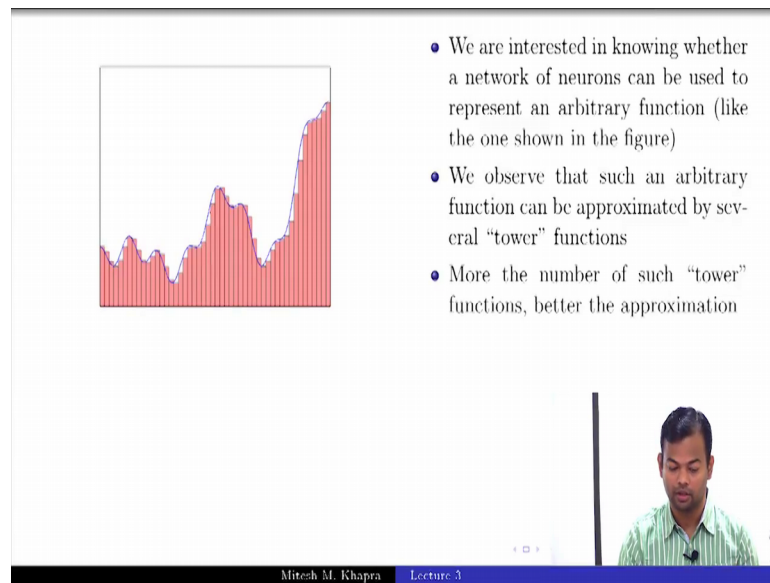


- We are interested in knowing whether a network of neurons can be used to represent an arbitrary function (like the one shown in the figure)
- We observe that such an arbitrary function can be approximated by several "tower" functions

Mitesh M. Khapra Lecture 3

Now, we make an observation that such an arbitrary function can actually be approximated by a lot of something that we call as tower functions, ok. These are all single I mean pulse functions which you have many of these, and you could have an approximation right and you can see that this approximation is bad at many places, right, but still it is an approximation it largely gives you the same shape as the original curve. What would happen if I increase the number of such tower functions?

(Refer Slide Time: 06:31)



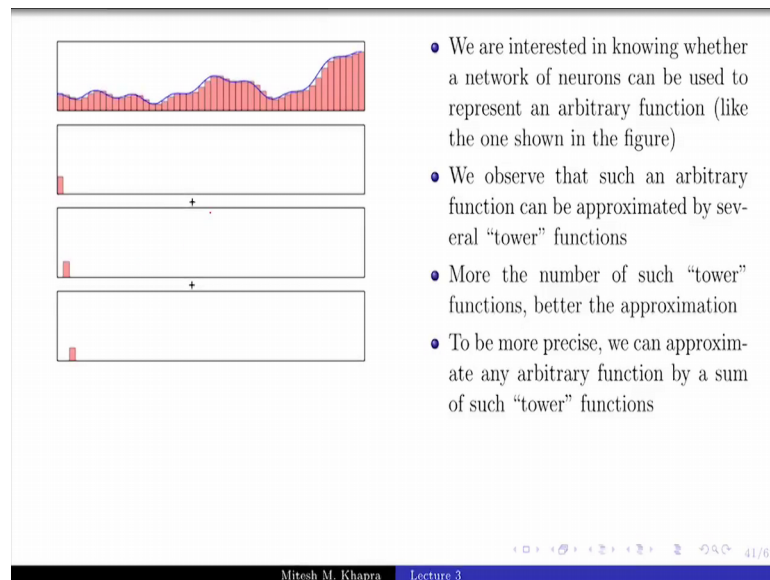
- We are interested in knowing whether a network of neurons can be used to represent an arbitrary function (like the one shown in the figure)
- We observe that such an arbitrary function can be approximated by several "tower" functions
- More the number of such "tower" functions, better the approximation

Mitesh M. Khapra Lecture 3

Student: (Refer Time: 06:29).

The approximation would improve, right. If I keep increasing it the approximation would go more and more better, right. So, now, just try to keep things in mind whether I write in the theorem, right you can make it arbitrarily close to the actual value; that means, you can keep doing something. So, that your approximation becomes better and better and you already see something of that sort. This is still in the sense of a figure, we need to relate this back to a neural network, but you see that as I am increasing these tower functions I become approx arbitrarily close to the actual function, ok.

(Refer Slide Time: 07:05)



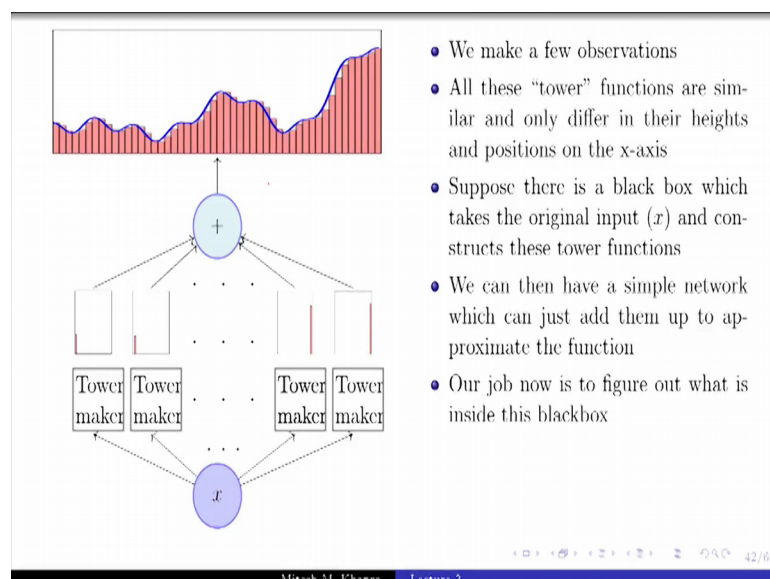
The slide shows a target function (blue line) and its approximation using a sum of three tower functions (red bars). The tower functions are simple, rectangular pulses of varying heights and positions. The approximation is shown as the sum of these three functions, with the resulting curve (red line) following the target function (blue line).

- We are interested in knowing whether a network of neurons can be used to represent an arbitrary function (like the one shown in the figure)
- We observe that such an arbitrary function can be approximated by several “tower” functions
- More the number of such “tower” functions, better the approximation
- To be more precise, we can approximate any arbitrary function by a sum of such “tower” functions

Mitesh M. Khapra Lecture 3 41/62

Now, this is what is actually happening right I have multiple such tower functions I am adding them up all of them are shifted in time rate. So, this tower function is actually this one this tower function is actually this one and so on, right and I have not drawn the remaining ones I am taking all of these tower functions adding them up and getting my original function, right and the more such tower functions have the better is the approximation. How many of you are perfectly fine with this? Ok good.

(Refer Slide Time: 07:35)



The slide shows a neural network architecture for function approximation. The input x is fed into a network of "Tower maker" blocks. Each "Tower maker" block outputs a tower function (red bar). These tower functions are summed together to approximate the target function (blue line).

- We make a few observations
- All these “tower” functions are similar and only differ in their heights and positions on the x-axis
- Suppose there is a black box which takes the original input (x) and constructs these tower functions
- We can then have a simple network which can just add them up to approximate the function
- Our job now is to figure out what is inside this blackbox

Mitesh M. Khapra Lecture 3 42/62

Now, you make a few observations right all these tower functions are actually the same what is the only difference they just shifted and their magnitude changes right, but they are all tower function right. So, let us think of this that if I know how to make a rectangle then I can make any rectangle, right. I just need to change the size of the rectangle and maybe shift it or oriented differently or something, right. So, they are all similar I just need to learn how to draw a tower right that is what my quest is.

Now, if I take the original input salinity pass it through multiple such blocks each block is capable of making a tower function and each of these would give me one of these towers that I am looking for and I am looking for so, many of these, right. If I have as many such tower makers then I could get these towers I could just add them up, and then get the original function back, right. And the more these I have the better is my approximation, right. So, I am taking as input the salinity and trying to predict the oil does this make sense, ok. Still we have not figured out a neural network way of doing this we are still building intuitions of how to do this, ok.

Now, our job now is to figure out what goes in this black box that is the tower maker and how does it connect to neural networks, if you figure that out then our story is complete. Then we know that a neural network can actually do this and that precisely proves the statement which I had made that it can it can represent arbitrary functions, alright. So, we will figure this out over the next few slides.

(Refer Slide Time: 09:12)

- If we take the logistic function and set w to a very high value we will recover the step function
- Let us see what happens as we change the value of w

$$\sigma(x) = \frac{1}{1 + e^{-(wx+b)}} \quad w = 50, b = 0$$

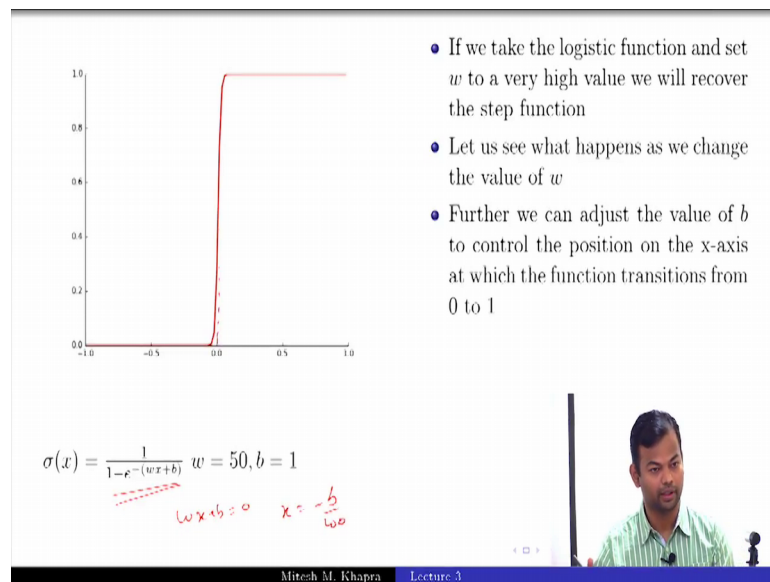
Mitesh M. Khapra Lecture 3

Now, if you take the logistic function and set w to a very high value, what will we get? Just try to think about it. The answer is already written, but I want you to imagine it. w covers what?

Student: (Refer Time: 09:31).

The slope, right; as I make w very high what will happen is I will get the. So, let us try changing the value of w , ok. I just increase the value of w and see what happens to the sigmoid curve.

(Refer Slide Time: 09:47)



Some error here, actually there is some problem the w value should have increased and that is how the sigmoid slope increases not the b value the b value comes later on, ok. So, actually sorry about this, the w value as I keep increasing. So, do not think that b is increasing think that the w is increasing. It will become sharper and sharper and it will come very close to the step function, right. It will not become exactly the step function that will only become in the limit, but if I keep increasing I will get very close to the step function everyone agrees with this, ok.

Now, what happens if I increase the value of b , it will shift everyone is confident about that, can you tell me why?

Student: (Refer Time: 10:34).

What will shift actually; the point at which the transition happens, right. So, what is this point actually?

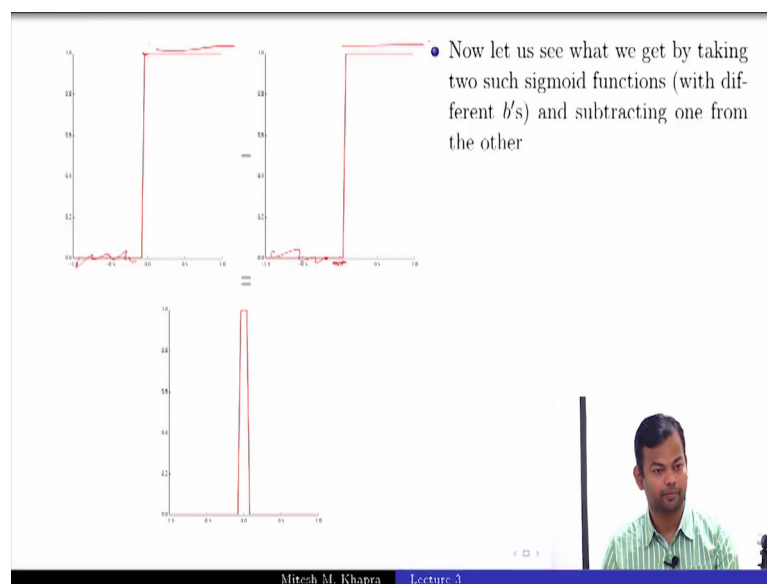
Student: (Refer Time: 10:43).

This is the point at which I get that half value, right and let us look at our function, this is a function. When will I get that half value? When $w x$ plus b is?

Student: 0.

0, right. So, that means, x equal to minus b by w , that is why it is proportional to b . So, as I keep increasing the value of b , this will keep shifting, ok. Is that fine everyone with this?

(Refer Slide Time: 11:16)



Now, what if I take two such modified sigmoid functions which are shifted differently and both are very close to the step function, right. So, here is where one threshold is, here is where the other threshold is and now I subtract this one function from the other, what will I get?

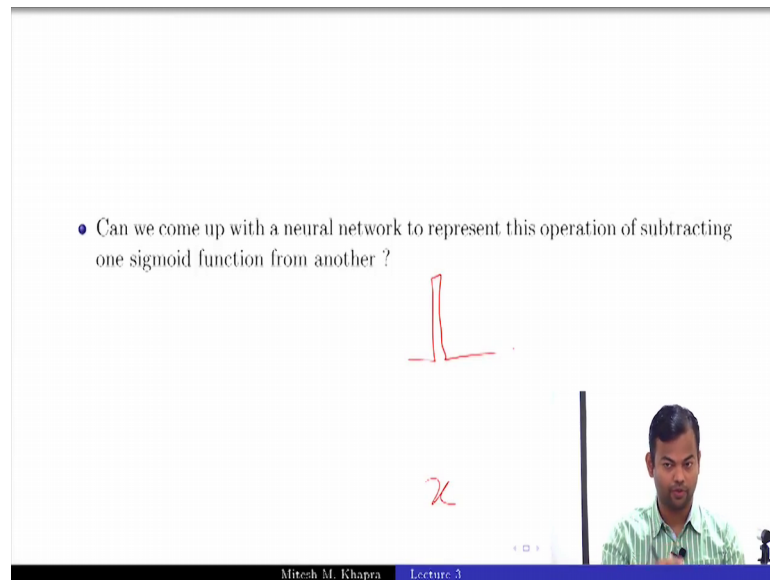
Student: (Refer Time: 11:35).

You know the term.

Student: Tower.

You will get a tower, right. Is that fine, everyone gets this, right? So, these places up to this point both are 0. So, 0 minus 0 will be 0, at this for this small range this is 1 and this is 0. So, that 1 minus 0 and then afterwards both are 1. So, 1 minus 1 would be 0. So, you get that tower function, ok. So, now I have my tower maker, ok.

(Refer Slide Time: 12:05)



Now, can we come up with a neural network to represent this operation? I want a sigmoid neuron, I was working with a sigmoid neuron with some arbitrary weights, right, so that I recover that step function, can you imagine? Now, given x , I want this tower function and that is exactly what one of the blocks was, right. So, what I am asking you is oh god,. So, I am asking you to give me a neural network for this. Can you think of it? Can you try imagining it?

Student: (Refer Time: 12:46).

Sorry?

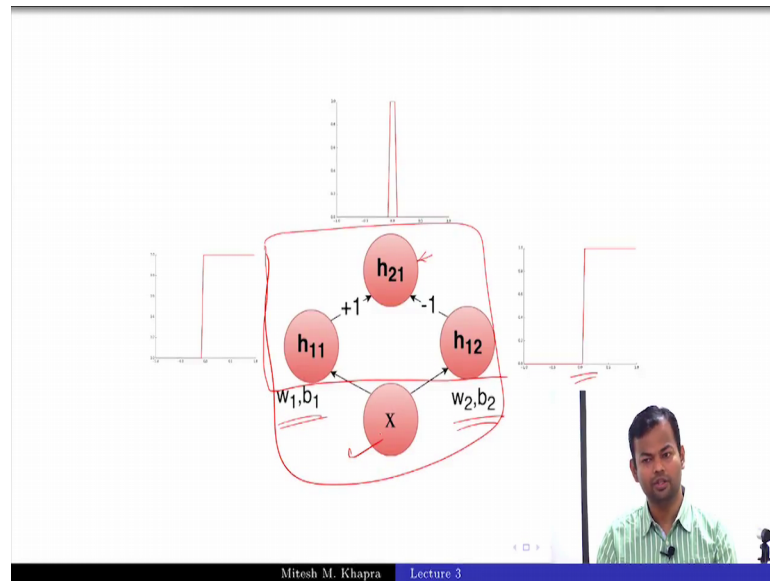
Student: (Refer Time: 12:48).

Two neurons in the hidden layer; how many of you agree with that? Ok, can you can you take some more time to imagine what it would be?

Student: (Refer Time: 13:00).

And I have already, right.

(Refer Slide Time: 13:07)

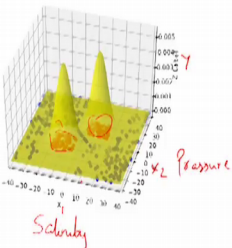


So, this w_1, b_1 if I set it appropriately I will get this step function. If this w_2, b_2 , I set it appropriately I will get this step function. Now, I needed to subtract one from the other, right. So, I will do plus 1 minus 1, this is just a simple addition and I will get this, is that fine? Everyone agrees with this? This is just an adder, right, this is just an aggregator everyone gets this? So, now, I have given you the tower maker? If you put enough of these tower makers and learn the w 's appropriately what will you get?

Student: (Refer Time: 13:48).

That function that we were needed. So, you can approximate it arbitrarily to any precision that you want as long as you keep increasing the number of these units, right. So, these units actually give you one tower more of these units that if you have actually this much this is the input ring, ok. The more such tower makers that you have the more is the bars that you will get and then you can approximate everyone gets the intuition behind this? Fine, ok. This all is always good in one dimensions.

(Refer Slide Time: 14:21)



- What if we have more than one input?
- Suppose we are trying to take a decision about whether we will find oil at a particular location on the ocean bed(Yes/No)
- Further, suppose we base our decision on two factors: Salinity (x_1) and Pressure (x_2)
- We are given some data and it seems that $y(\text{oil|no-oil})$ is a complex function of x_1 and x_2

Mitesh M. Khapra Lecture 3

Now, what will happen in two dimensions? What if we have more than one input? What is the tower there? Do you do you guys all do all know what is the tower there?

Student: (Refer Time: 14:32).

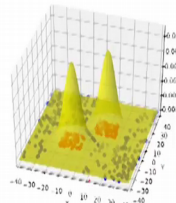
If you say no, I will give you a zero on the assignment. Remember the last question of the assignment? Did you all make a tower? Did you all make a 2D tower? Did you all copy that? No. So, what if we have more than one inputs suppose you had again trying to take a decision about whether we will find oil at a particular location of the ocean, right and suppose, now we base it on 2 two, right. So, say this is salinity, this should be x_1 , should be x_2 should be y , and this is pressure, ok.

Now, just observe about the red and blue points. So, the red points are where you will not, for those configurations of salinity comma pressure you will not find oil and the blue points are for which you will find oil. What is the one thing that you can tell me about the red points and the blue points? Not linearly separable, right, but we still want to be able to learn this, is that fine, a single perceptron cannot do it I will also make a case for a single sigmoid neuron cannot do it and then I will show you that. In fact, first I will show you that with a network of neurons we can do it and then I will show that: with a signal single sigmoid neuron you cannot not actually do that.

So, now this is again a valid problem you could have we could imagine that you will get this kind of data where you have two factors and your function is some arbitrary function of these two factors. It is not a need linear boundary between the blue and red points. Everyone sees that the blue and red points are not linearly separable you cannot draw a plane such that all your red points lie on one side and the blue points lie on the other side, everyone sees that ok, but the solution which I have plotted here that is a good solution. It makes sure that all the red points are in this region and the blue points are outside.

So, it will predict a high value for these red points and a zero value everywhere for the blue points, is that obvious? How many of you understand that figure? Ok, good.

(Refer Slide Time: 16:46)

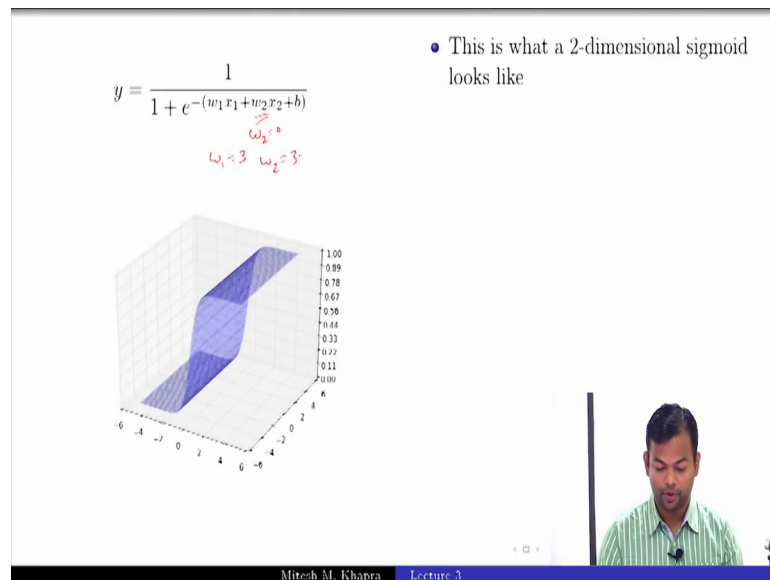


- What if we have more than one input?
- Suppose we are trying to take a decision about whether we will find oil at a particular location on the ocean bed(Yes/No)
- Further, suppose we base our decision on two factors: Salinity (x_1) and Pressure (x_2)
- We are given some data and it seems that $y(\text{oil|no-oil})$ is a complex function of x_1 and x_2
- We want a neural network to approximate this function

Mitesh M. Khapra Lecture 3

So, now I want to show that even in 2- dimensions I could come up with arbitrary. I could come up with a neural network which could actually approximate this and again what will I look for? A tower maker, right. I just want something which can make towers and approximate it, ok.

(Refer Slide Time: 16:48)



So, this is what a 2-dimensional sigmoid looks like slightly incorrect because I have what I have done is I have actually said w_2 to 0. So, if you actually I would want you to do this go back and plot this for w_1 equal to 3 and w_2 equal to 3, ok.

Just go back and plot this and see what. You get you will not get such a smooth such a nice looking S, but you will still get something which looks like looks like a snakes hood, right. So, in still get that S shaped function it just that it would be bent at some points and it be thinner at some points and broader at the other points. So, just go back and see and then you will realize what is happening, right.

(Refer Slide Time: 17:29)

$$y = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + b)}}$$

$w_1 = 25, w_2 = 0, b = 45$

- This is what a 2-dimensional sigmoid looks like
- We need to figure out how to get a tower in this case
- First, let us set w_2 to 0 and see if we can get a two dimensional step function
- What would happen if we change b ?

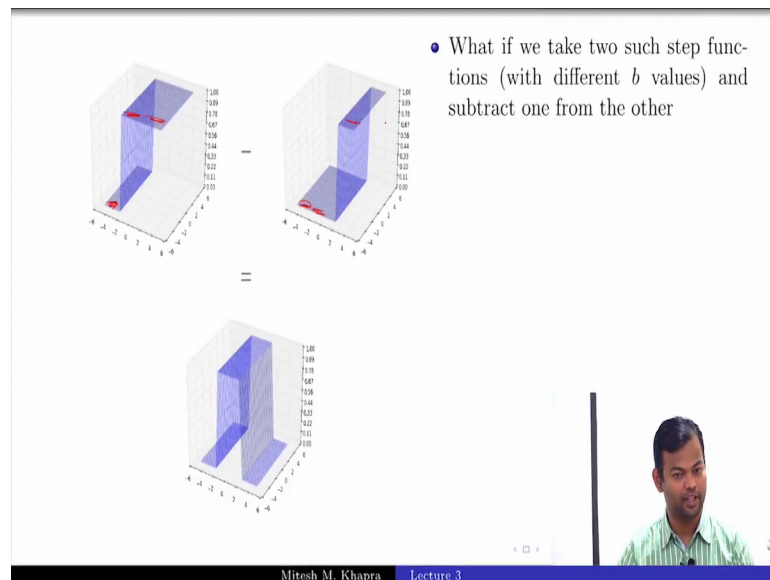
Mitesh M. Khapra Lecture 3

So, here again what we want to figure out is from the single sigmoid I was able to take you to a tower function, right. From a 2-dimensional sigmoid what does a tower mean here and how do I take you to the tower? Ok. So, that is what I want to do. So, I have said w_2 equal to 0 and I will it will become obvious why I have done that. So, just understand what the figure is doing, right. So, this if you just look at this is like the cross cut, right so, you are looking at the front view of this figure and that is just the sigmoid function without the w_2 right and now, since I have said w_2 equal to 0, no matter what I set x_2 to the same function will get repeated throughout, that axis, do you get that?

So, that is why this entire function is just getting repeated throughout this axis and then you just get a similar S shape function, everyone gets that? How many of you do not get that? How many of you get that? Ok. So, this if you look at the front view this is the sigmoid of one variable, but since I have said w_2 to 0, no matter what I change x_2 to, the function is going to remain the same. So, it will just get copied throughout the x_2 axis, is that, fine with you.

Now, what will happen if I increase w_2 , sorry w_1 ? Same thing right, it will just keep shifting till it becomes almost like a 2D step function, ok. Now, what will happen if I increase b ? Shift. I can do the same thing here also; same logic applies here also, ok.

(Refer Slide Time: 19:12)



Now, what is the next step that I am going to do? Take two of these which are shifted by some point and, then subtract what will I get? Everyone had this figure in mind? So, just see right. So, this portion both are 0, so, 0 minus 0 would be 0 this portion this is 1, but this is 0. So, that would be 1 minus 0 and again in this portion both of them are 1. So, 1 minus 1 would be 0. So, you will get this kind of function would you like to live in such a tower? I am very serious, yes or no? No? Why?. It is open from two sides, right. You cannot live in this tower. So, you want something which is a closed tower, right. So, how will you do that give me an intuition?

Student: (Refer Time: 19:51).

We will do the reverse thing. What will be set to 0?

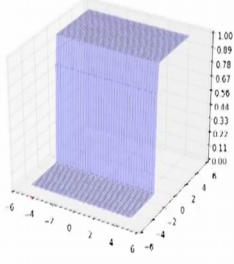
Student: (Refer Time: 19:57).

W 1, ok.

(Refer Slide Time: 19:58)

$$y = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}$$

- Now let us set w_1 to 0 and adjust w_2 to get a 2-dimensional step function with a different orientation
- And now we change b



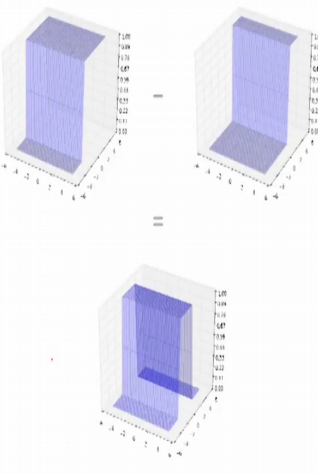
$w_1 = 0, w_2 = 25, b = 20$

Mitesh M. Khapra Lecture 3

And, this is how it would look the orientation would change and again so, notice that this is your sigmoid function and since I have set x_1 to 0, no matter what I change along the x_1 axis the same function gets copied and you get a nice looking a sigmoid function, ok.

Now, again I will do the same thing I will increase the w , I will get a close to a step function I will increase the b . I will move along this axis.

(Refer Slide Time: 20:24)

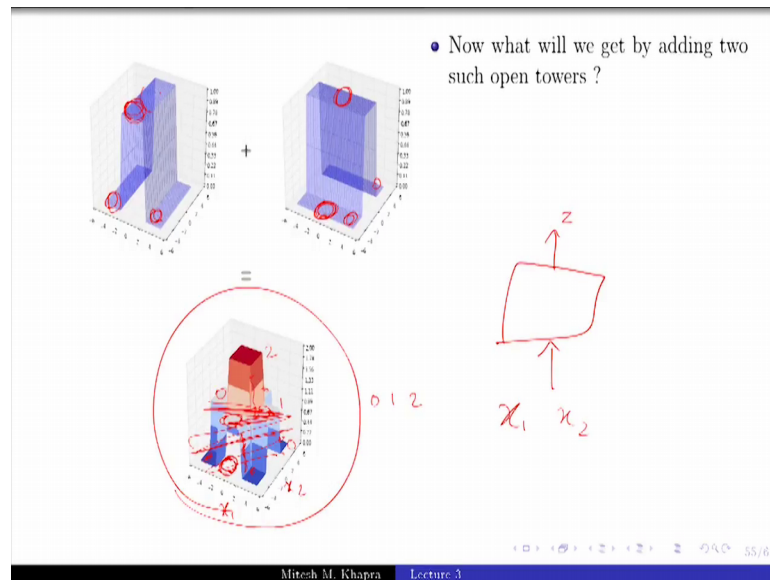


- Again, what if we take two such step functions (with different b values) and subtract one from the other
- We still don't get a tower (or we get a tower which is open from two sides)
- Notice that this open tower has a different orientation from the previous one

Mitesh M. Khapra Lecture 3

Next step, take two of these subtract get what? Another tower function, this is also not a tower that you like to stay in. So, what do I do now? Add them, sure? Add this tower to the other tower; this is what you did on the assignment? You do not remember, ok, fine.

(Refer Slide Time: 20:44)



So, now what will happen if I add these two, will you get a tower function? What will you get?

Student: (Refer Time: 20:51).

You will get a tower function with a parking floor, right? Is that what you will get? Everyone understands why this is? So, these portions both are 0. So, you get 0 same logic applies for all the four corners, right, is that fine; now, for this portion or rather this area, right. So, this guy is 0, this guy is 1. So, you will get a 1, the same logic applies for all these four corners in the centre both are actually 1. So, 1 plus 1 would give you 2. So, this is 2, this is level 2, this is level 1, this is level 0, is that fine?

So, what am I done so far? I have taken my x_1, x_2 passed it through some transformations, right this what are these transformations we will see, but transform it through these multiple hoops, right where I adjusted a w 's and b 's and I have got some z right and this is how that z behaves. For different values of x_1 comma x_2 , I will get these different values and these values range from 0 to 1 to 2, is this pictured clear? I have taken my original x_1 comma x_2 , passed it to some of these transformations and

irrespective of what my x_1 to x_2 is this tells me the entire range of values that I will get. For some combination of x_1 comma x_2 I will get 0, for some combinations I will get 1, for some combinations I will get 2 and some combinations also between 1 and 2, right. So, these places where it transitions, is that clear, is that picture clear to everyone? Ok.

So, now I can treat this as the output z , ok. Now, from here how do I go to a tower?

Student: (Refer Time: 22:47).

Threshold it; how will you threshold it? What do you want? You only want this much part to exist, right this without the parking floor. How will you do it? Any output which is greater than equal to 2, you want to keep it any output which is less than 2, you want to make it 0. If I do this will I get a tower? Right, sorry, greater than equal to 1 any value which is greater than equal to 1, you want to keep it, anything which is less than 1 you want to make it 0. So, this entire thing will get demolished. How do you do this? This is an if, else, ok. What?

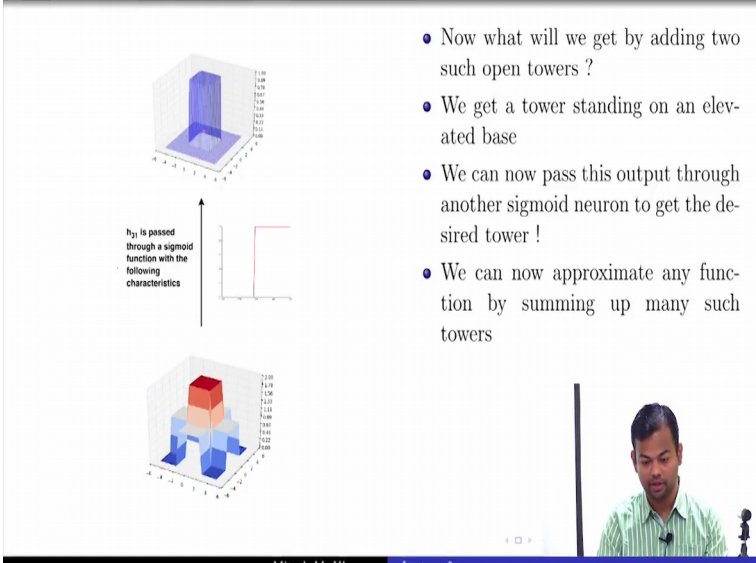
Student: (Refer Time: 23:40).

If else if something is greater than equal to 0, do something else do something else, what is that?

Student: (Refer Time: 23:55).

Perceptron, right, but we do not want to use perceptron. We want to use sigmoid neurons. Have you learned an approximation from a sigmoid neuron to a perceptron. Very high w , right; you get the intuition. Let us see what we do on the next slide.

(Refer Slide Time: 24:07)



• Now what will we get by adding two such open towers ?

• We get a tower standing on an elevated base

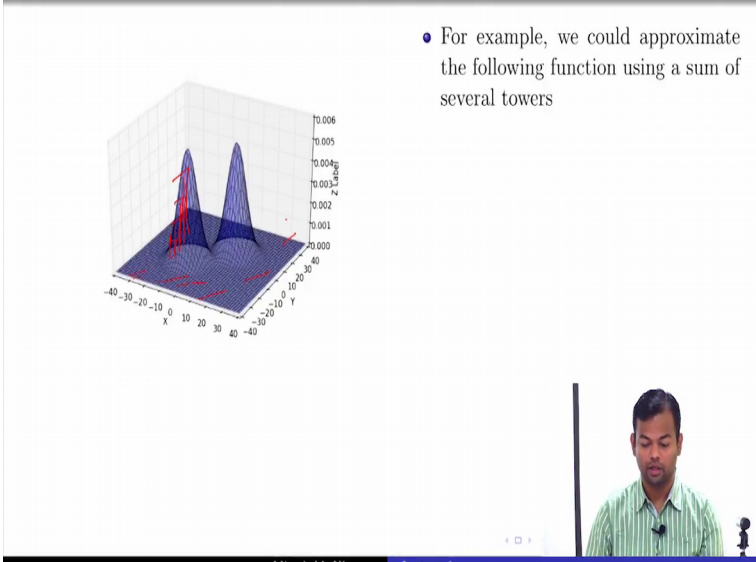
• We can now pass this output through another sigmoid neuron to get the desired tower !

• We can now approximate any function by summing up many such towers

Mitesh M. Khapra Lecture 3

So, I take this any z which comes from here I will pass it through a sigmoid neuron which are very high slope such that the threshold is at 1, anything which is greater than 1 will pass out as 1; anything which is less than 1 will go to 0. So, everyone sees how we took this structure and converted it to a tower. We have this tower now, now what do I do with this?

(Refer Slide Time: 24:34)



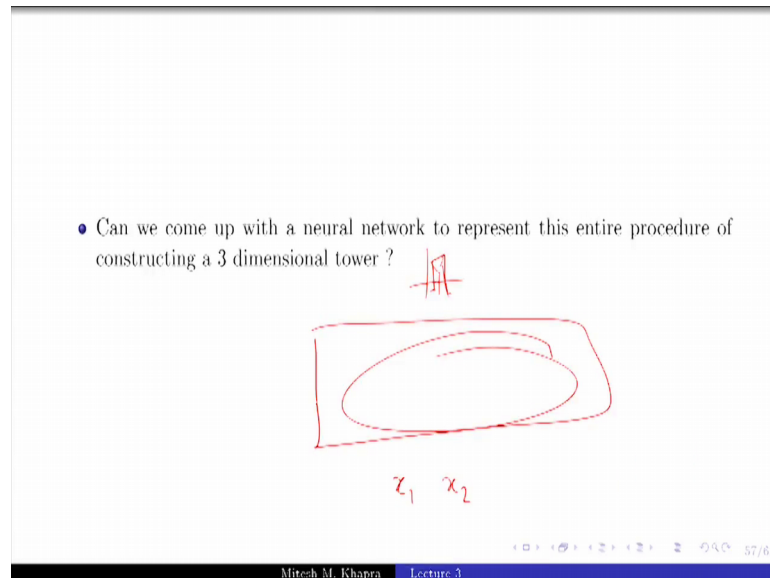
• For example, we could approximate the following function using a sum of several towers

Mitesh M. Khapra Lecture 3

I lead multiple such towers and I can approximate this I could put a tower here and so on. I could have these multiple towers and here of course, all my towers would be of 0

height right in this region, right. So now, I can cover the entire 2D space with a lot of tower functions and approximate this exactly, that is a very weird statement, approximate this exactly I mean approximate this to arbitrary precision, everyone gets this? Do you see why we constructed these tower functions and now we can put them inside this cone and approximate it.

(Refer Slide Time: 25:09)

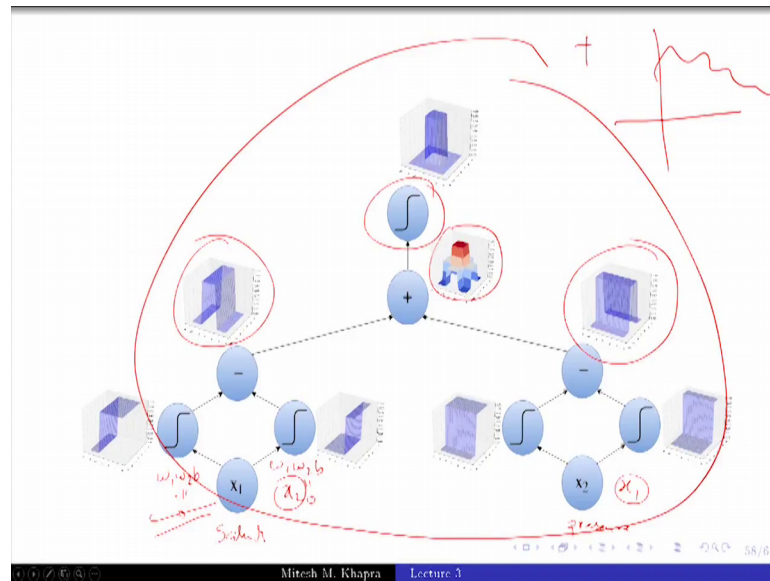


Now, all this is fine I was making some towers there. So, can you now give me a complete neural network which does this? I want you to imagine that. Remember you are taking what I am asking you to do is this x_1 , x_2 give me this such that I get this 2-dimensional tower I do not know how to draw it, something like this maybe whatever. So, I want this 2-dimensional tower what is this network of perceptrons going to look like? Just go back to all the operations that we did and try to imagine in your mind.

Student: (Refer Time: 25:47).

No, we will not use perceptron because we can always use a sigmoid neuron instead of a perceptron with the high w . I do not expect you to answer this I just want you to imagine; right, we just try with a there is something known as a pen, there is something known as a paper, ok. So, here is the solution.

(Refer Slide Time: 26:08)



So, what is happening here you have this salinity and oh actually this is slightly wrong I do not know why you guys saying it is correct. Actually, at both places I need both the inputs it is just that in one case I do not care about that input because I have said $w_2 = 0$. So, I learn these weights w_1, w_2, b , w_1, w_2, b of course, here the network should learn that w_2 is equal to 0, right and then you get this one tower do not needs this to be modified, this figure is incorrect.

So, we need x_1, x_2 both as inputs we need to label it with w_1, w_2 equal to 0 and b and so on it. So, we will discuss this later, anyways, but you get the idea right that you take these two inputs make one tower, take the inputs again make another tower, add them up to get this function, pass it through this step neuron function step sigmoid function, so that you get the tower. So, this is one block. You will have many such blocks each of which will learn different w 's and b 's. So, that they get shifted and then you will place them all together you have an aggregator on top of this which will combine them. Just a minute, how many of you get this? Good.

Student: (Refer Time: 27:31).

Yes. So, that is a good question, I am going to come to that, right. So, I have very conveniently given you a solution where I have what is the bad thing that I have done? I have hand coded these things, right. I have hand coded w_1 's, w_2 's and b 's, is that

fine in practice? No, I mean that is where we started off and we do not want to hand code these, right.

So, now you know a learning algorithm for a single sigmoid neuron. Now, what you have is a network of neurons right for this network of neurons, I need to give you a learning algorithm driven by the objective function that whatever output it gives would be very close to that arbitrary function that you are trying to model.

If I give you a learning algorithm then you would be convinced that if this has to be minimized and the weight configuration which need it needs to arrive at as w_2 is equal to 0, then the algorithm should be able to do that, right. Because, we saw we have some faith in these algorithms in the case of a single sigmoid neuron that with the right objective function it will give me a principled way of reaching that objective function. In this big network my objective function is to arbitrarily to approximate this of this true function, right.

So, now if I give you that as the objective that whatever outputs the network generates; so, the network might generate something like this. So, that has to be very close to the true output that is the objective function that I am going to use in that learning algorithm and if that learning algorithm works which will prove then you should be able to arrive at the necessary weights to make this approximation, right, is that clear and in fact, there might you might not even have to do these multiple towers in practice. All I am trying to prove is that there is one solution which exists, right.

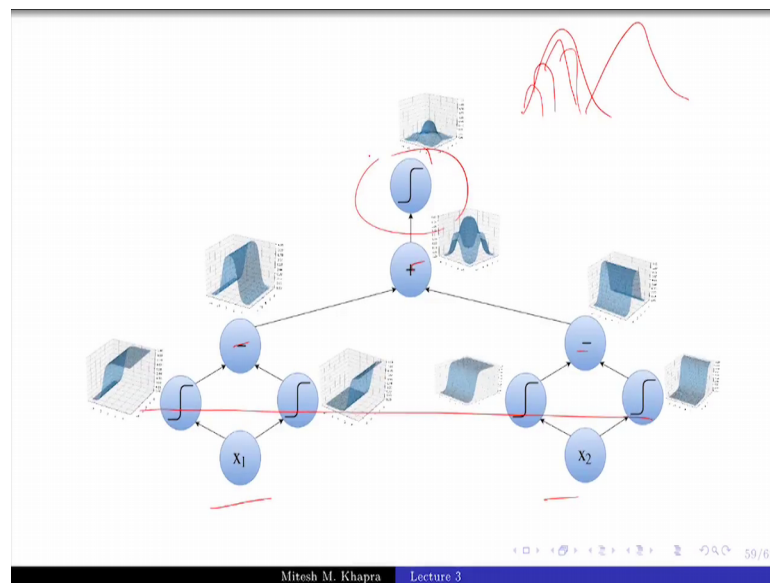
If there is one solution which exists I can say that locate the network can learn, that is the only claim I make. I am not saying this is the only solution, right. Same as in the case of the Boolean functions, where I said that one solution exists where you have to raise to n neurons of the hidden layer that was a sufficient solution, that was not a necessary solution for the AND function we were actually able to do it with a single sigma neuron, right. So, just keep that in mind I am just giving you a sufficient solution.

And, the network could actually learn something better than this all right this is again a very bulky solution, why? It scales with the number of neurones' proportional to number of input variables that you have. So, that is for a sufficient solution, but you would want something better than that. All I am trying to say is that it can approximate I am just

telling you the representation power and just as we had the catch there that the hidden layer is very large the same catch applies here also, is this story clear to everyone?

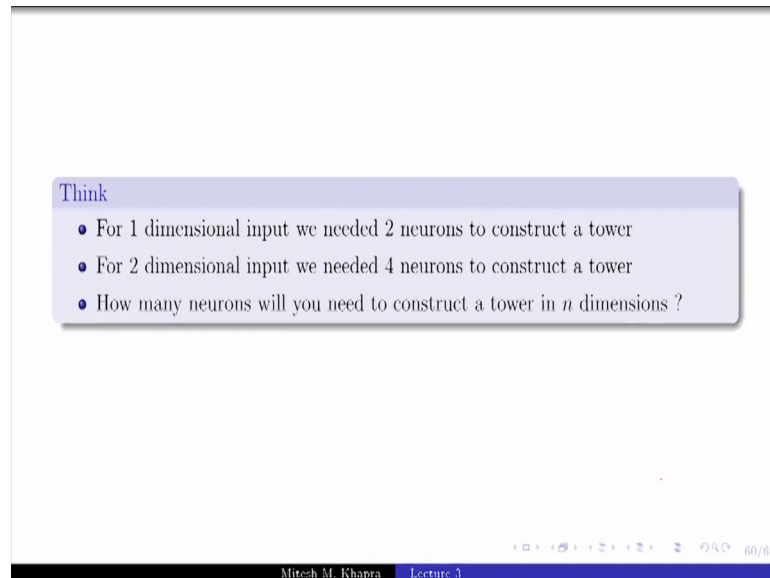
So, I have given you a solution I have not told you how to learn the weights I have given you a network. Now, later on we will discuss a learning algorithm for this network. And we will have some confidence that given a particular objective function that learning algorithm can strive to go to minimum error or minimize the quantity of that objective function that is going to come in two lectures from now, is that fine?

(Refer Slide Time: 30:32)



And, that was for the tower function now; I could have actually directly done this right. So, I wanted to approximate these functions. So, I could have placed a lot of these kinds of things here and approximated it, right. So, that instead of that very high slope sigmoid function I could just use a normal sigmoid function also. And again there is a error here, but I hope you get the picture, it is just that you feed both the inputs to them.

(Refer Slide Time: 30:56)



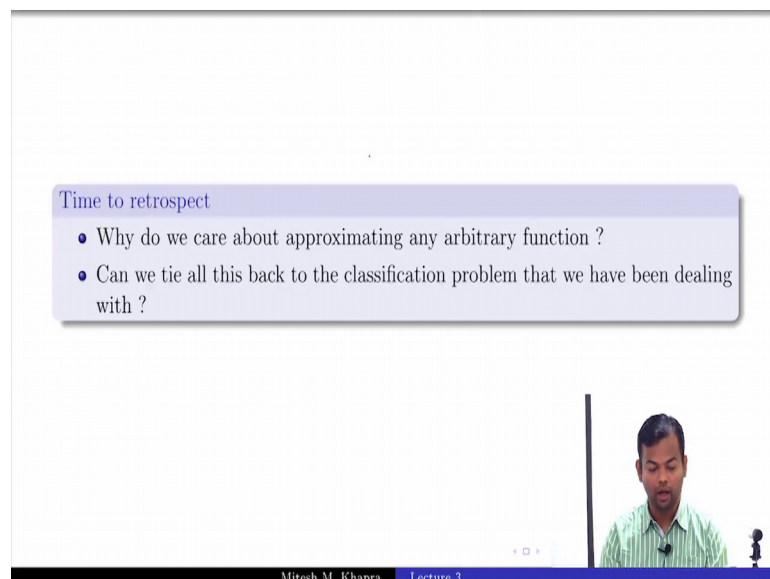
Think

- For 1 dimensional input we needed 2 neurons to construct a tower
- For 2 dimensional input we needed 4 neurons to construct a tower
- How many neurons will you need to construct a tower in n dimensions ?

Mitesh M. Khapra Lecture 3 60/62

So, for 1 dimensional input we needed 2 neurons to construct a tower. For 2 dimensional input how many neurons did we need? I am just counting these because these are simple aggregators, right and this is one constant at the end. So, how many did we need actually $2n$, I mean $2n$ of I mean. So, for n how many would we need? Let us try to work that out, ok. So, I will ask you that in the quiz how many do we need for n dimensions.

(Refer Slide Time: 31:30)



Time to retrospect

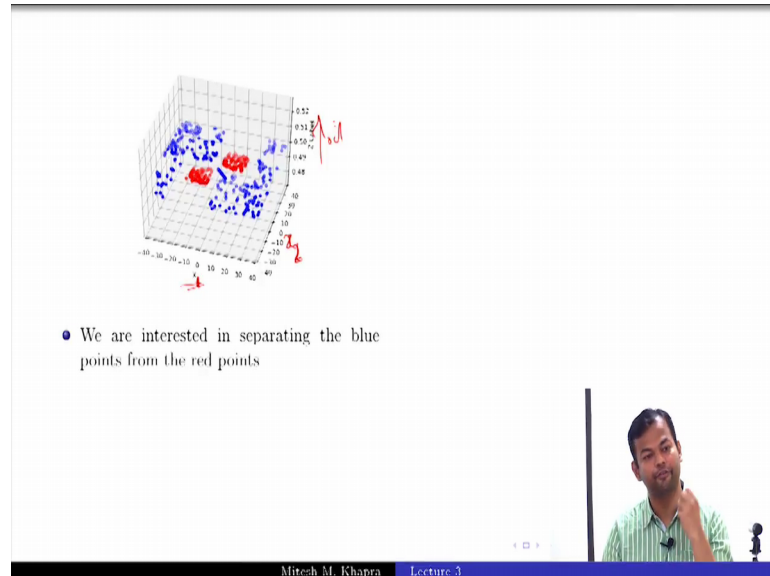
- Why do we care about approximating any arbitrary function ?
- Can we tie all this back to the classification problem that we have been dealing with ?

Mitesh M. Khapra Lecture 3

Now, why do we care about approximating any arbitrary function? We will again try to close the loop now, we saw that we can arbitrarily we can approximate any arbitrary

function. But now again I want to come back to the point why do we want to do this and can we tie this back to the classification problem that we were dealing with.

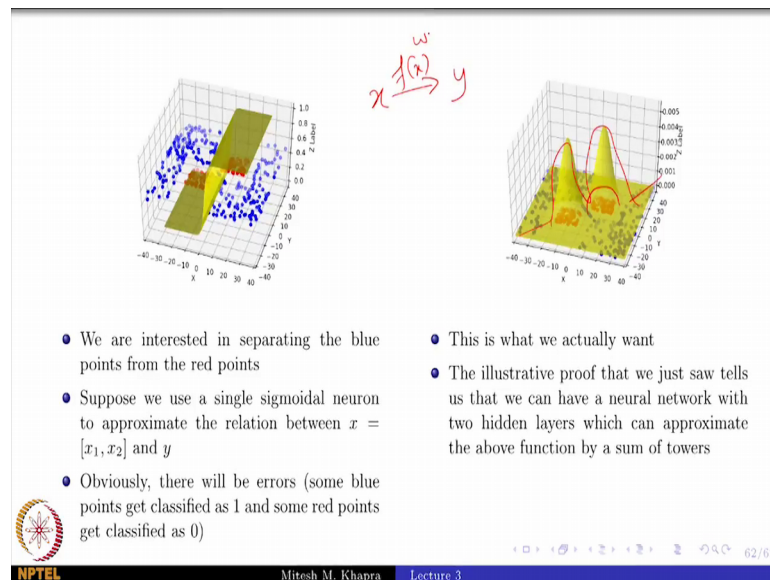
(Refer Slide Time: 31:47)



And, this is the data which I had given you which was there were some points some values of x and y sorry; this should be x_1 and x_2 it is, where this is pressure and salinity or salinity tendency and this is the output which is oil, ok.

Now, there was this is what the function actually looks like now what would have happened if I had used a single sigmoid neuron to try to approximate this function try to represent this function and sigmoid neuron in 2 dimensions, right, so, the 2 dimensional sigma what would have happened? Can you give me one solution for this? Remember earlier I had said that perceptron cannot handle data which is not linearly separable, but then I anyways used it for data which was not linearly separable. And we got some line such that we got some errors the red points and the blue points are not clearly separated. So, I am asking you for a similar thing here, I force you to use a sigmoid neuron, what would you give me?

(Refer Slide Time: 32:47)



Is this fine? This is one of the possibilities of course, it could have been oriented differently and several things. What is happening here is that for these blue points it is acting correctly, but for these red points it is not acting correctly. I am assuming red is positive and blue is negative, I think that should have been the other way round, but let us assume red is positive and blue is negative. Again, now for these red points this part is working fine, but it is misclassifying all these blue points.

So, all these bad locations is actually saying that you can find oil and for all these good locations here it is saying that you cannot find oil, that is what a sigmoid neuron would do and you could have multiple solutions are possible here right, but all of them would have this problem that will make errors on some red points and some blue points right, but the true solution that we wanted is something like this. Again there are multiple solutions possible, right. You could have anything is an error you could have even finer one side you could just have this much there many things possible this is one such solution. What the illustrative proof told you is that, you can actually use a network of perceptrons and approximate this arbitrary function which exists between the input variables and the output variable.

So, if this is the function which exists between the input variables and the output variables now, you could take these multiple 2 dimensional tower functions and approximate it. With the catch that you might need many of these in the hidden layer, but

you can still do that, ok. So, that is why this in theorem important because now any problem that you take right any problem that you will have in machine learning would always want you to take an x learn a function of x which takes you to y this function will be have some the function will have some?

Student: (Refer Time: 34:30).

Parameters, right and now what this theorem is saying is that you could adjust these parameters such that you can arbitrarily come close to the true function, right. So, that is the significance of this. Any machine learning problem that you can think of in the sense of classification or regression you would find that this is useful and I am giving you a very powerful tool to do that. Of course, with the catch that I am not giving you any bound on the number of neurons that you will need, I am just saying use as many as you want, right, is that fine?

So, that is where we will end today's lecture.