

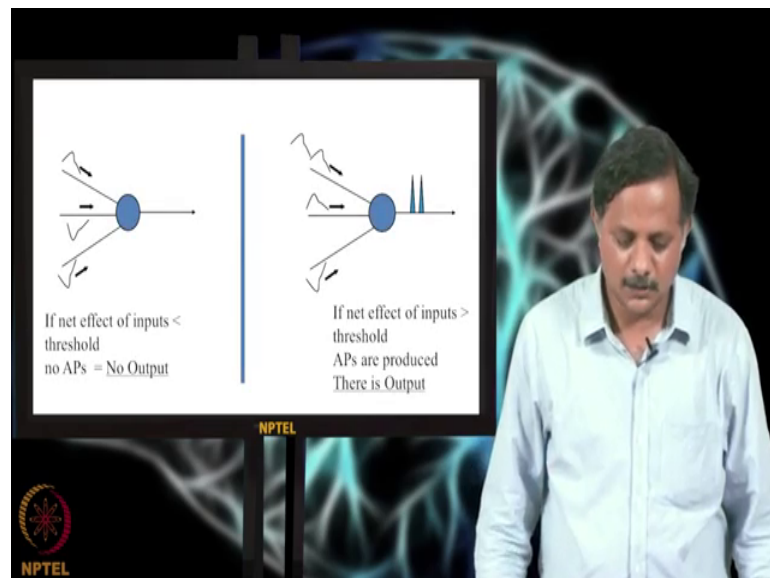
Demystifying the Brain
Prof. V Srinivasa Chakravarthy
Department of Biotechnology
Indian Institute of Technology, Madras

Lecture – 07
Networks that Learn - Segment 1

Welcome to the lecture 4 segment on of the Course Demystifying the Brain. In the last lecture, we talked about Neurons and Signaling. We discussed what kind of signaling process occur in the neurons, we talked about how signals flow down the dendritic tree and gets summated in the cell body and action potentials are generated in the in the axon hillock and then how the property down the axon and now arrive at the synapse and communicate and how these signals was communicated across the synapse towards the neurons.

Now, in this lecture, we will talk, we will construct networks with this neuron brain like networks and see what kind of interesting things this networks can learn.

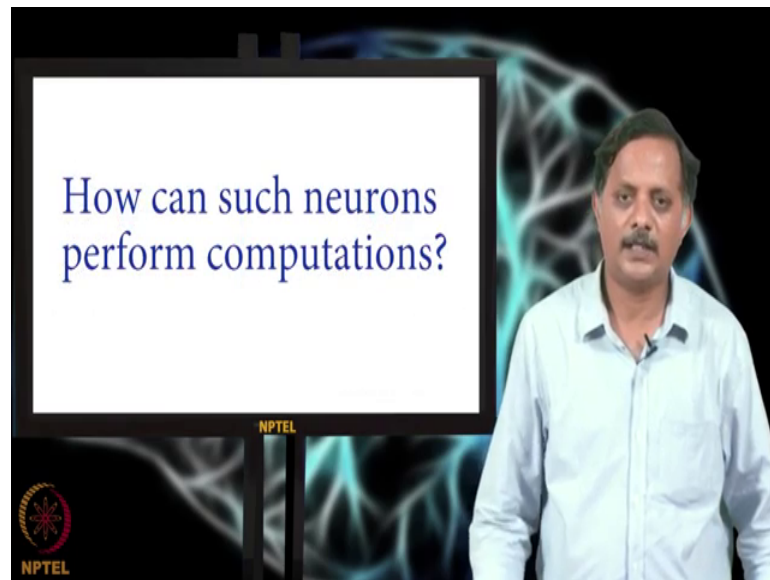
(Refer Slide Time: 00:54)



So, in the last lecture, we looked at simple neuron model like this. So, where I have a neuron and results inputs by generate some other neurons. And on the in the figure on the left you see one e p, s p flowing down on the top dendrite and two e p, s p s flowing along on the low dendrites. All these things get added up in soma and then since negative is a dominate there is no action potential generated therefore, there is no output.

In the figure on the right there are two e p s p s, in the top most dendrite and then one e p s p in the middle one and one I p s p in the bottom one. When all of them add up, the positive is dominate and therefore, the at the soma potential crosses of threshold and if action potential generation and there is some output.

(Refer Slide Time: 01:51)



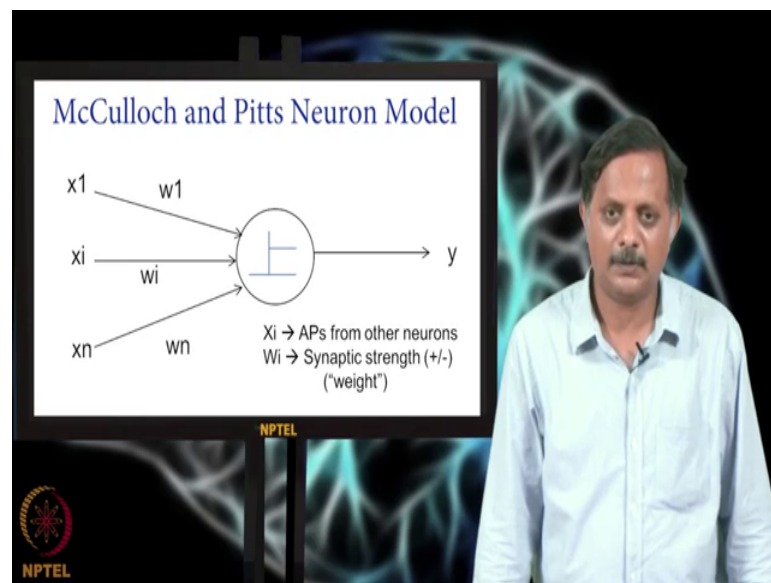
Now, given that a single neuron waves like this, if you construct a network of this neurons and think of you know correct network as the brain like network, what kind of a computations can such network perform?.

(Refer Slide Time: 01:57)



So, this was the question that was asked by these two people, way back in 1940s. So, they were Warren McCulloch, who was a Neuro Scientist and Walter Pitts that who was a Mathematician. These two people got together and then asked the question right what kind of computations can be done with the neuron model that you have just looked at.

(Refer Slide Time: 02:14)

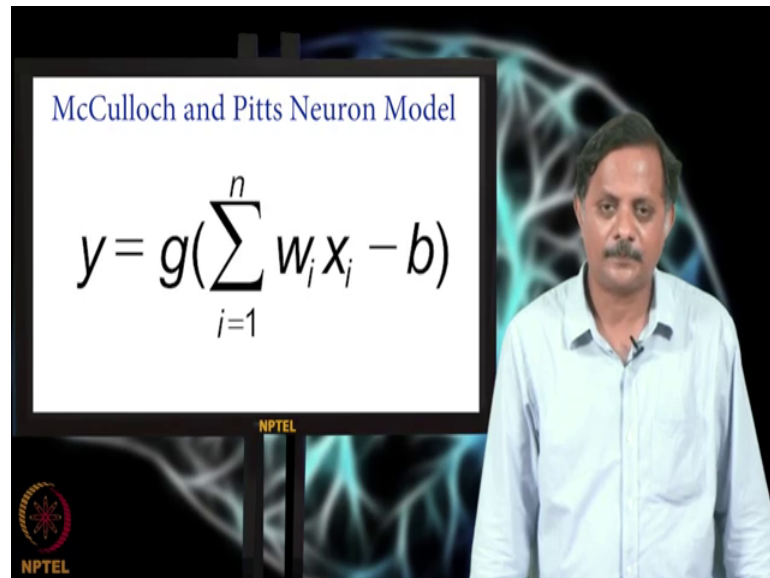


So, the intuitive neuron model, which you have described verbally until now in the last lecture. The McCulloch and Pitts have taken that kind of a neuron model and formulated a mathematical equation right to describe this such a neuron. This see neuron model which is called the McCulloch and Pitts neuron model he is described like this. So, this neuron receives inputs from other neurons these inputs are like are denoted by x_1 , x_2 and x_n ; that means, there are n inputs that are received by this neuron. And x_1 , x_2 you can think of them as action potential is coming from other neurons. Then the quantities w_1 , w_2 and w_n represent the synapses between these other neurons. And the neuron that you are looking at in the center w actually represents the synaptic strength right and we have seen in the last lecture the Synaptic strength can be high or low and can be positive or negative.

So, w can take values which are high or low and can be positive or negative. And this quantity is called weight in Computational Domain. So, you have all these weights and so, when all these inputs are received by the neuron they get added up with appropriate

weight by w_i after multiplying with appropriate weight and that is denoted by the summation $\sum w_i x_i$.

(Refer Slide Time: 03:25)



McCulloch and Pitts Neuron Model

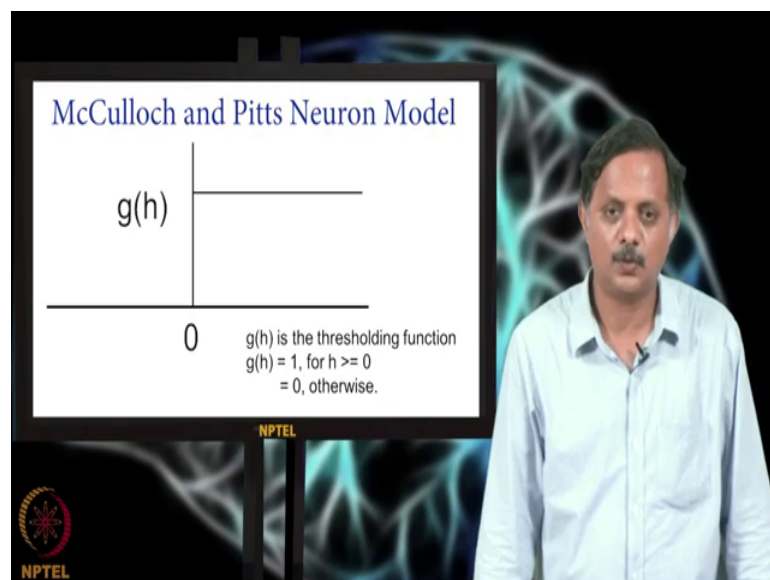
$$y = g\left(\sum_{i=1}^n w_i x_i - b\right)$$

NPTEL

The slide features a man in a light blue shirt standing next to a presentation board. The board displays the title 'McCulloch and Pitts Neuron Model' and the equation $y = g\left(\sum_{i=1}^n w_i x_i - b\right)$. The NPTEL logo is visible at the bottom left of the board.

And when the summation causes a threshold, which is denoted by b in this case then the output is non-zero, it is 1. And when the summation does not cross the threshold output is 0.

(Refer Slide Time: 03:39)



McCulloch and Pitts Neuron Model

$g(h)$

0

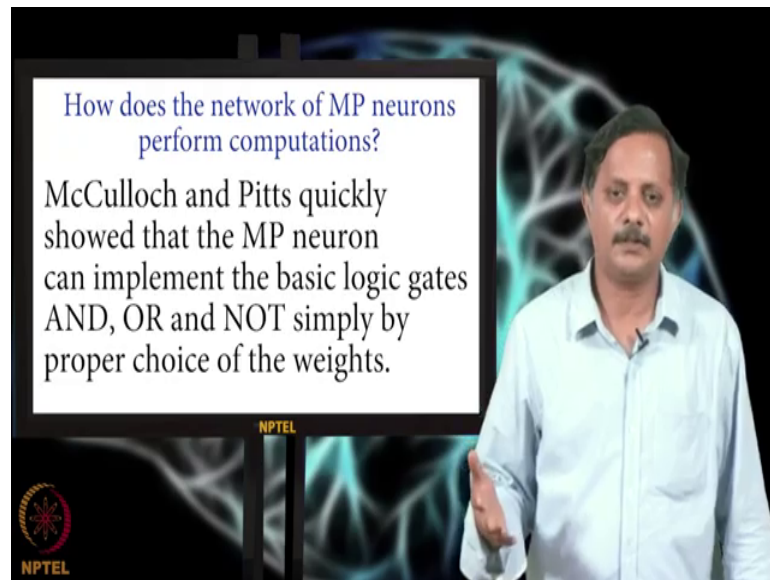
$g(h)$ is the thresholding function
 $g(h) = 1$, for $h \geq 0$
 $= 0$, otherwise.

NPTEL

The slide features a man in a light blue shirt standing next to a presentation board. The board displays the title 'McCulloch and Pitts Neuron Model' and a graph of the thresholding function $g(h)$. The graph shows a horizontal line at $g(h) = 0$ for $h < 0$ and a horizontal line at $g(h) = 1$ for $h \geq 0$. The NPTEL logo is visible at the bottom left of the board.

And this thresholding is achieved by the function g , which is basically the threshold function or what is called the step function where g of h is given as it is equal to 1 when h is positive or 0 and it is equal to 0 otherwise.

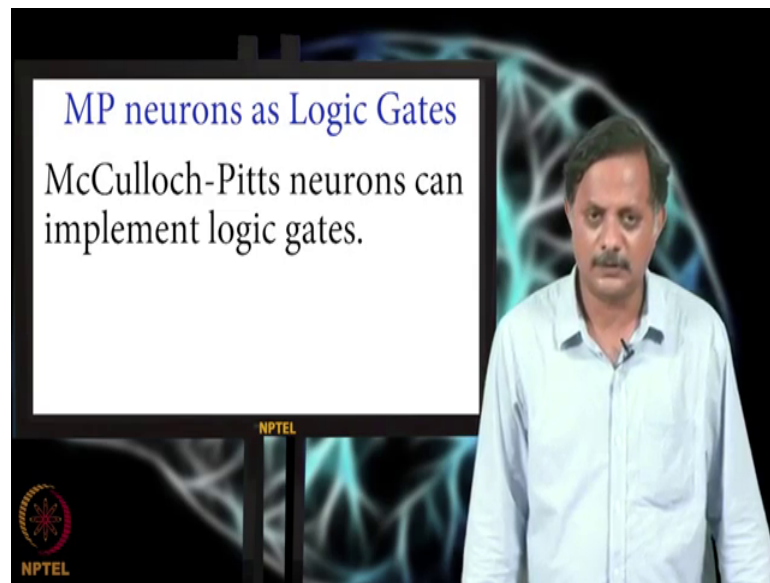
(Refer Slide Time: 03:56)



So, now, if you take this McCulloch and Pitts computation neuron models and construction networks out of them, what kind of a network when you construct involving the behave?

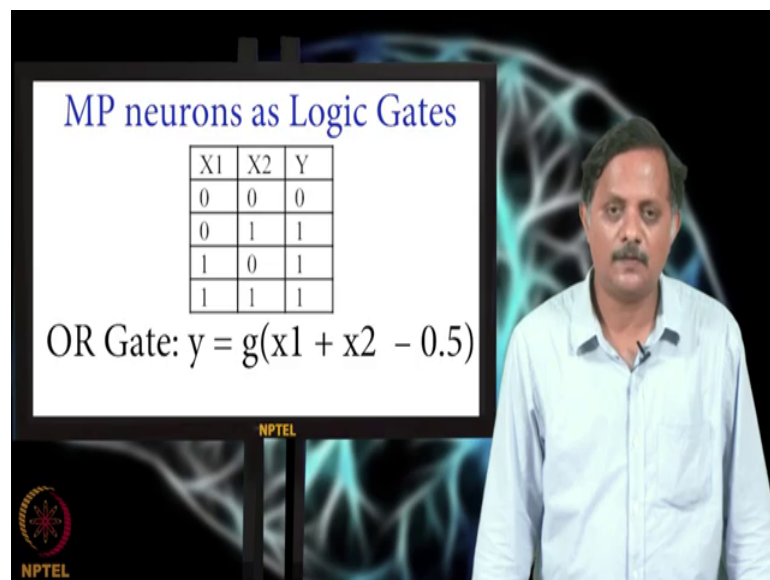
So, McCulloch and Pitts thought of the brain has a big logic circuit. So, because they have shown that the McCulloch and Pitt's neuron models can be made to behave like in the logic gates. So, as you know that modern computer is constructed out of Boolean logic. So, or Boolean circuits Boolean circuits are constructed out of 3 basic fundamental logic gates, the AND gate, OR gate and NOT gate. And by combining these gates in various ways, you can ca construct any desirable Boolean logic circuit.

(Refer Slide Time: 04:40)



So, therefore, McCulloch and Pitts have shown that the 3 fundamental logic gates can be constructed using the McCulloch and Pitts neuron model.

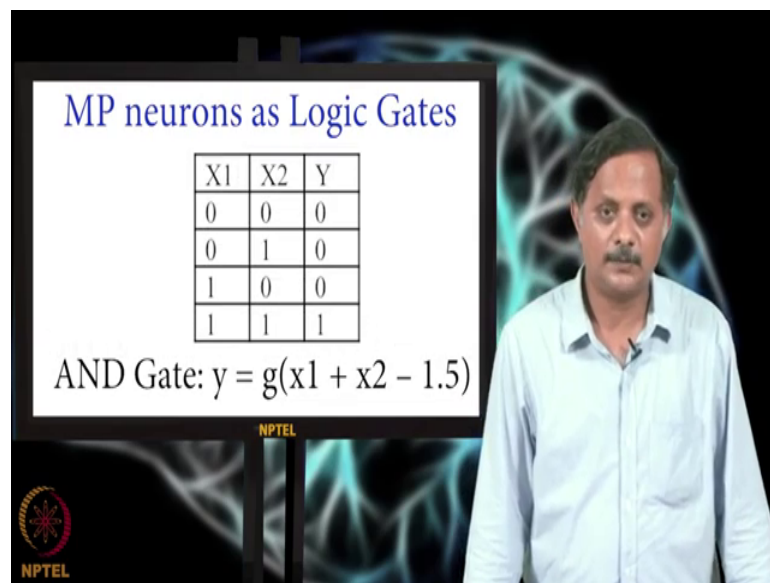
(Refer Slide Time: 04:50)



So, for example, if you want the OR gate, the OR gate truth table can be seen in the right top in the slide. So, the inputs are 0, 0, 0; 1 1 0 1 1. So, the inputs are 0, 0; output is 0 and inputs for input 0, 1, 1, 1, 0 now 1 output is 1 because if either x_1 or x_2 is equal to 1, then output is 1

So, you can get this logic gate with a McCulloch and Pitts neuron model, which has 2 inputs as shown in that in the formula here. So, y is equal to g of x_1 plus x_2 minus point 5. So, if x_1 x_2 are both zeros right, you have g of minus point which is 0. So, right under if we you even if 1 of x_1 and x_2 are ones right then the argument of g crosses becomes a positive value. So, therefore, y is 1. So, you can see that the McCulloch and Pitts neuron model in this case, behave like behaves like logic gate.

(Refer Slide Time: 05:48)



MP neurons as Logic Gates

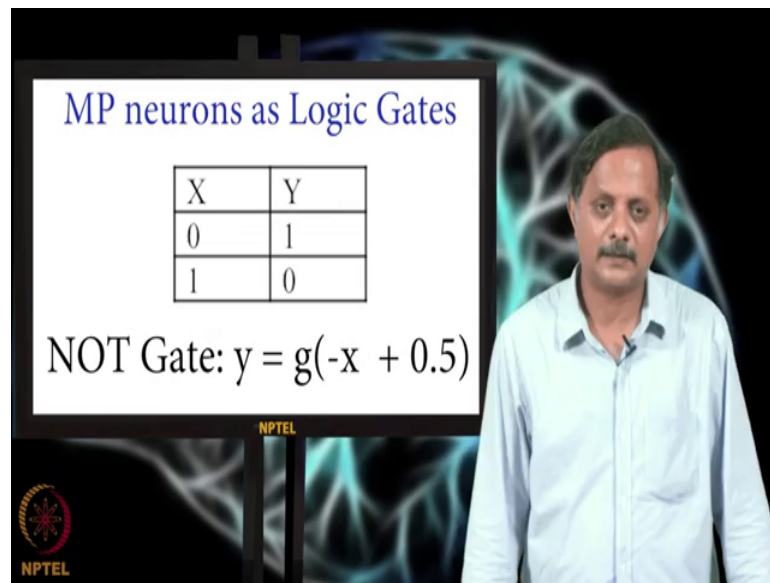
X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate: $y = g(x_1 + x_2 - 1.5)$

NPTEL

Similarly, in the AND gate, if you just increase a threshold from 0.5 to 1.5, it will easily verify that the argument becomes positive only when both x_1 and x_2 are equal to 1; that means, x_1 and x_2 are equal to 1 and otherwise the argument is negative. Therefore, y 0 when for the first 3 cases that is 0, 0, 0, 1, 1, 0 and 1, 1 and 1, 0 and y is 1 only when both x_1 and x_2 are equal to 1.

(Refer Slide Time: 06:20)



MP neurons as Logic Gates

X	Y
0	1
1	0

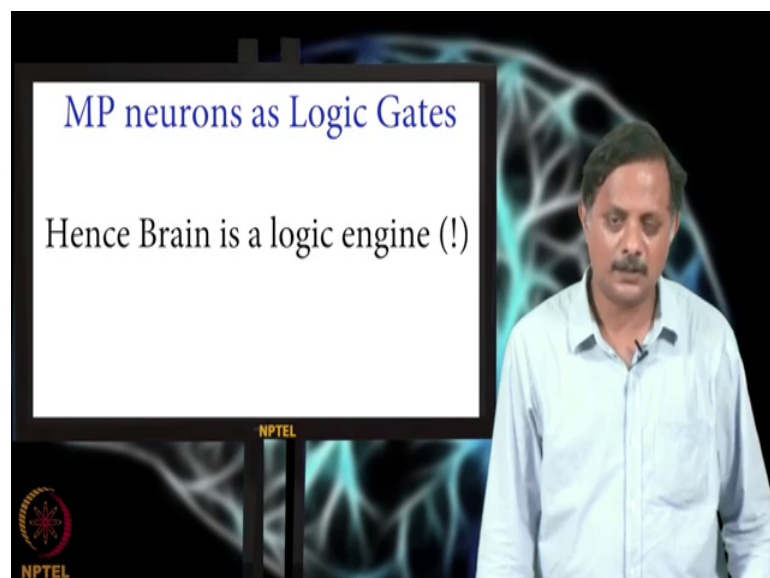
NOT Gate: $y = g(-x + 0.5)$

NPTEL

Then the third gate is NOT gate. So, in this case there is only one input x and there is only one output y . So, when x is 1 y should be 0 and when an x is 0 y should be 1. And you get that you can verify that very easily that the single this in neuron model which has only single input right behaves like logic gate.

So, therefore, they could not they showed how you can construct more complicated Boolean networks using McCulloch and Pitts neuron models and they have concluded that Brain is a logic engine.

(Refer Slide Time: 06:49)



MP neurons as Logic Gates

Hence Brain is a logic engine (!)

NPTEL

So, this kind of a thinking was even fueled by the social and historic background right at that time; because they worked in the early part of the 40s, 1940s and that was the time when the world war was going on and that was the time when people were exploring the use of computing technology you know extensively for war applications or military applications and in as a part of that kind of an effort large scale effort right.

(Refer Slide Time: 07:20)



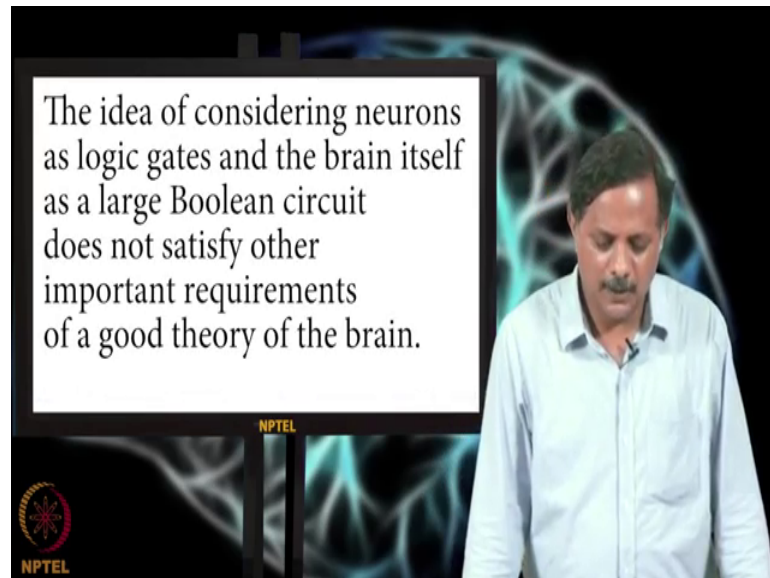
Ah this huge computer, that you can see in this picture. A computer that fills an entire room right was built in and commissioned in 1946.

This computer was called Eniac and it was the ka constructed in university of Pennsylvania. It was used for lots of military applications like calculating the project of you know missiles and you know cracking enemy codes and things like that.

So, with all this since going on the background, it was very natural that McCulloch and Pitts thought of brain is a computer; because they could see all the power of the modern computer or at least modern by the standards of 1940s and that it was itself was quite overwhelming, because at that time people were manually calculating the you know the this trajectories of missiles and that what took 20 hours you know to do the calculation with this Eniac computer was able to solve in only 30 seconds. So, it could really see the power of the computer.

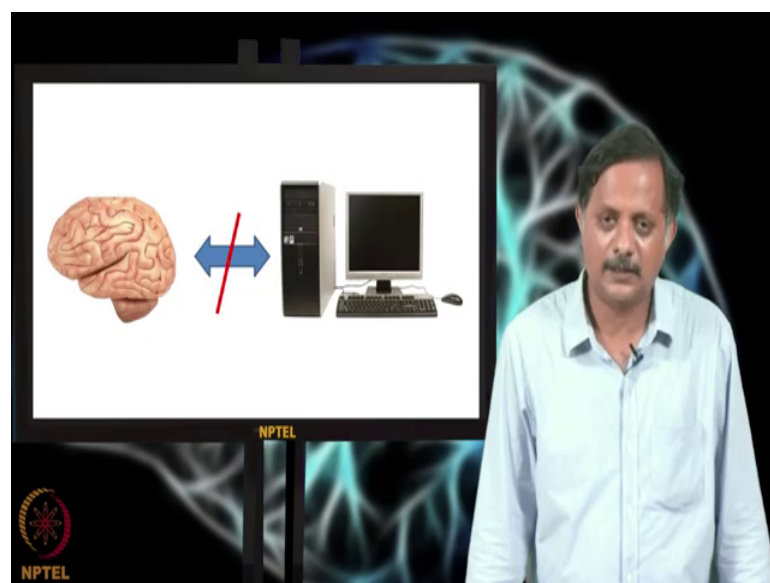
So, they thought right brain also must be like a computer and right and neuron is like a logic gate.

(Refer Slide Time: 08:21)



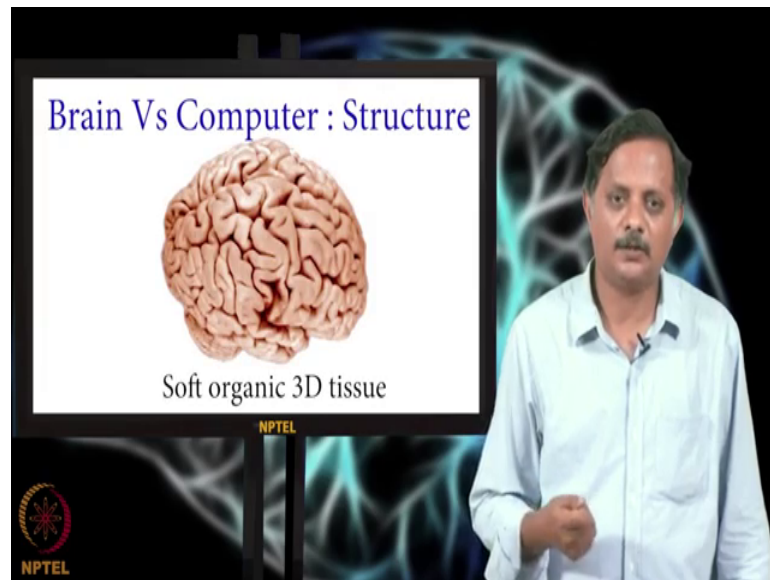
But point is say it is a very interesting idea it is very insightful, but unfortunately if you looked at more data from brain, you would very quickly realize that brain is not like a big logic circuit or a Boolean circuit. And the analogy between the computer and brain cannot be taken too far, why is that?

(Refer Slide Time: 08:29)

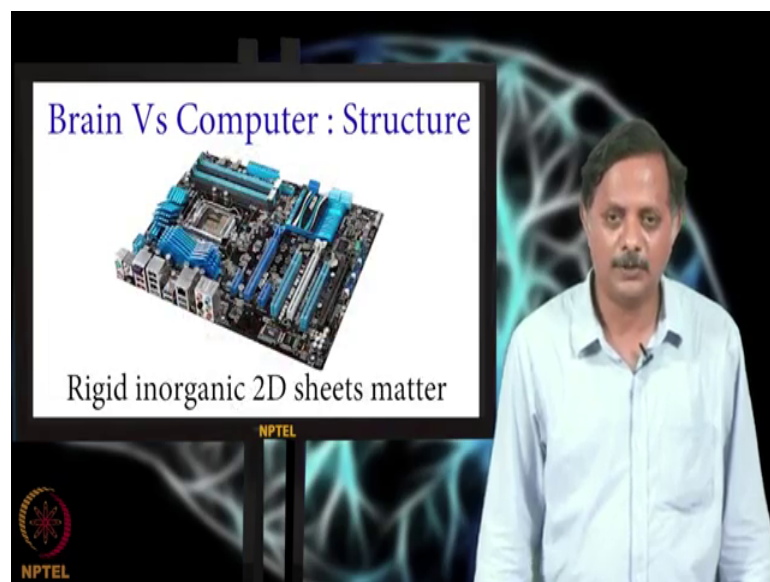


Let us look at some several properties right and compare brain with a computer. First of all structural in terms of the material that is used right brain is a soft organic tissue it is a 3-D volume right mass of tissue and it is wet and it is warm ok.

(Refer Slide Time: 08:48)



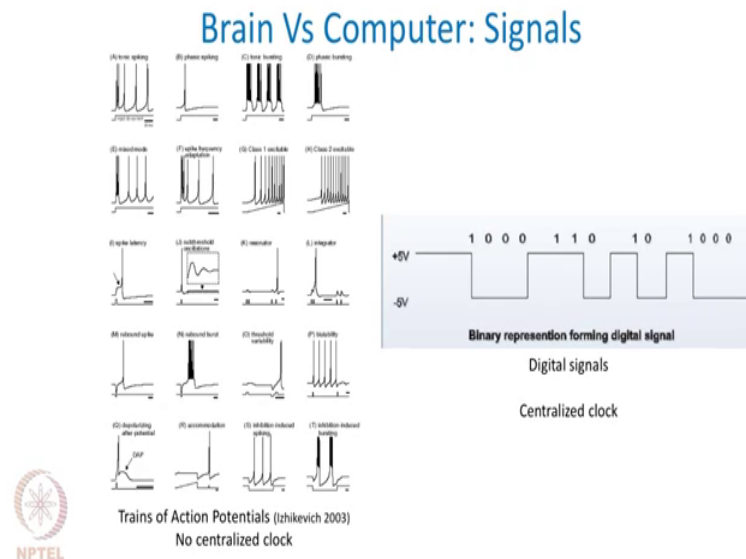
(Refer Slide Time: 08:58)



And whereas, the computer, if you look at a mother board for example, is a rigid hard inorganic 2 dimensional sheets of matter and also if you look at the chips which are housed in various various chips inside write the VLSI circuits themselves, the 2 dimensional sheets of matter.

So, in that sense there is a lot of difference. So, now, if you look at the signals that are used in a brain and as suppose to computer right, we have seen that in the neuron signals are action potential. the trains of action potentials. The actual firing patterns vary from neuron to neuron there are there is a there are a wide variety of neurons in terms of that can be classified in terms of their firing properties.

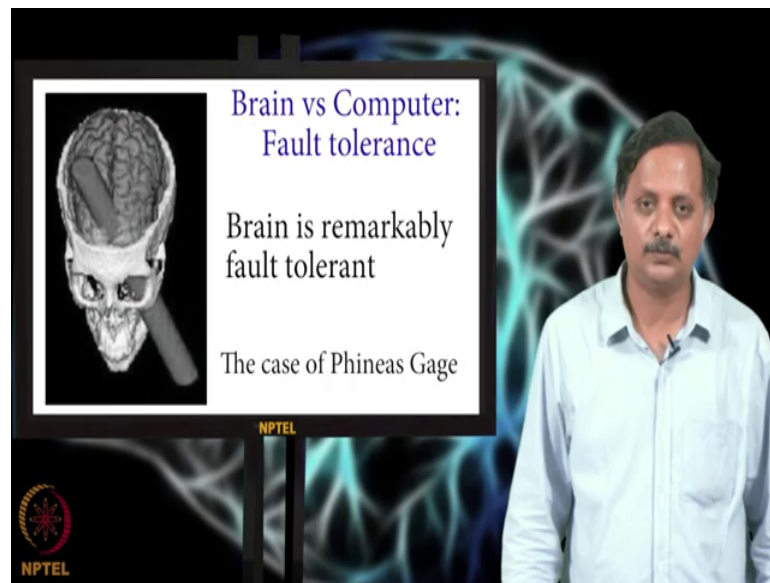
(Refer Slide Time: 09:35)



So, you can see some of these properties firing patterns in the figure on the left, but if you look at the computer and if you just swap any or bus or in a computer and look at the signal, it will look like simple rectangular wave going up and down and going between 2 voltage limits. So, minus 5 volts and plus 5 volts or something like that right. So, the in terms of signals used both Brain and Computer are very different.

As even more fundamental difference between brain, computer where philosophically there is a fundamental difference in the design philosophy of this 2 systems which is Fault tolerance. Now, to illustrate that let me give a small historical note.

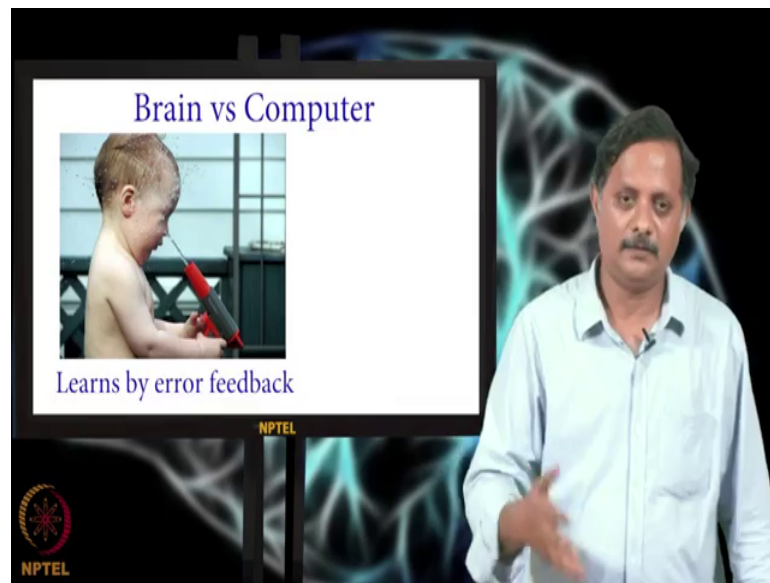
(Refer Slide Time: 10:17)



So, this incident took place in the late 19th century in the US. So, there was this guy called Phineas Gage who was a rail road construction worker, his job is to burry bombs and burry explosive dynamite in soil and soften the soil and break the rock and things like that. So, he was doing that one day and he was packing the emanation in a whole in deep whole using a crowbar and unfortunately the bombs went off exploded right on his face and the crowbar went through this head and actually, what you are looking at in this picture is a reconstruction of his skull.

So, 3 that crowbar went through his brain through the you know frontal lobe of the brain and remained like that and most miraculously the person survived that accident. He was taken to a nearby hospital and the crowbar was removed, surgery was performed and he survived that surgery. There was some personality failures, but what is quite interesting is the person survived. Now, imagine a similar accident right that occurs to a computer and you imagine you are driving a metal rod through the chassis of a desktop. I mean that will be the end of that desktop right. So, brain is remarkably Fault tolerant whereas, a computer is not, does not take very kindly to you know these kinds of faults. There is another very important different screen brain computer which is a brains or we humans right learn, so, we learn by error feedback right.

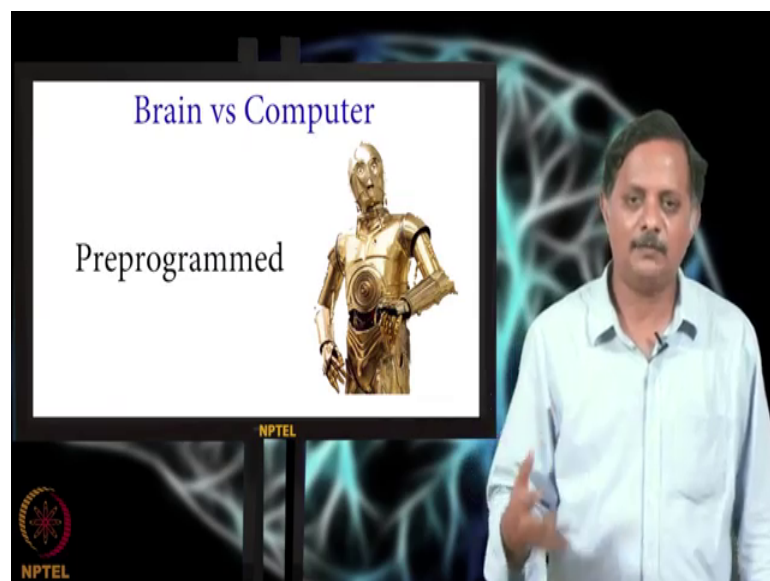
(Refer Slide Time: 11:44)



You want you make a perform an action in a in under certain condition then some in sub context and look at what the environmental show in that context. And then, if it is good then probably you do the same thing over again next time and if you get a negative feedback in the environment you stop doing it.

So we learn by trial and error no by getting feedback in the environment.

(Refer Slide Time: 12:07)



Whereas, a computer is preprogram and especially some of the older ones like this see 3 p o from Star wars like these are all like programmed machines the modern machines

you know with a lot of inputs from AI and machine having that different story, but otherwise traditionally computers are preprogrammed in a rigid fashion.

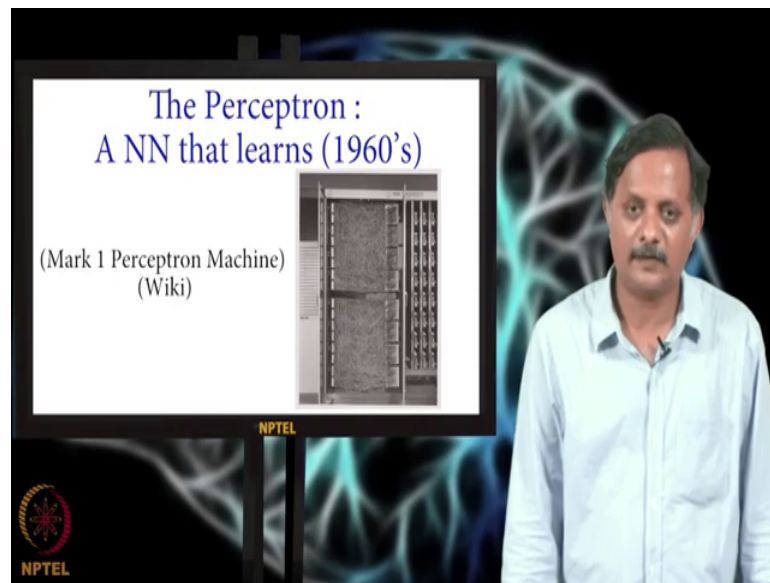
So, so, the therefore, this was not a very fair comparison to compare brain with a logic circuit and with a computer and think of the brain has a network of logic gates just like a Boolean network. So, this relation prompt at this person called Frank Rosenblatt.

(Refer Slide Time: 12:47)



You know in the late 50s to use the same McCulloch and Pitts neuron model, but construct a different kind of a network. Now, he showed how you do not have to hard code the weights the way McCulloch, Pitts have done whereas, in case of McCulloch Pitts they what they have done is they have just set the weights of various neuron. So, that they behave like an and gate and or gate and things like that. Whereas, in Rosen blatts what Rosen blatts has achieved is he showed you can train a neuron right and make it perform some task. And he called this network a Perceptron because it is able to perceive and classify patterns, visual patterns.

(Refer Slide Time: 13:23)

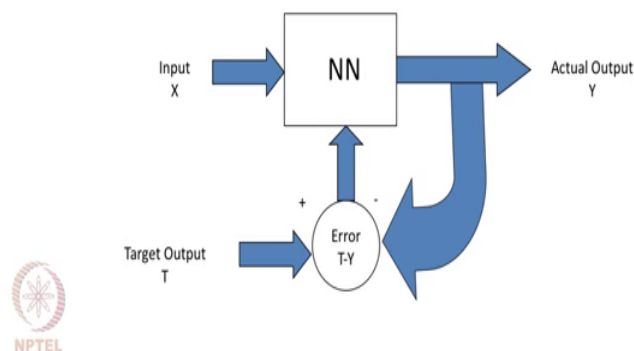


So, on the right hear and on the slide you see a kind of a retinoic implementation of the perceptron machine, it is called Mark one. So, basically per perceptron learns by error feedback just like you know we do just like brains do.

(Refer Slide Time: 13:37)

Learns from Experience

- NN can learn the unknown I/O relationship from sample data

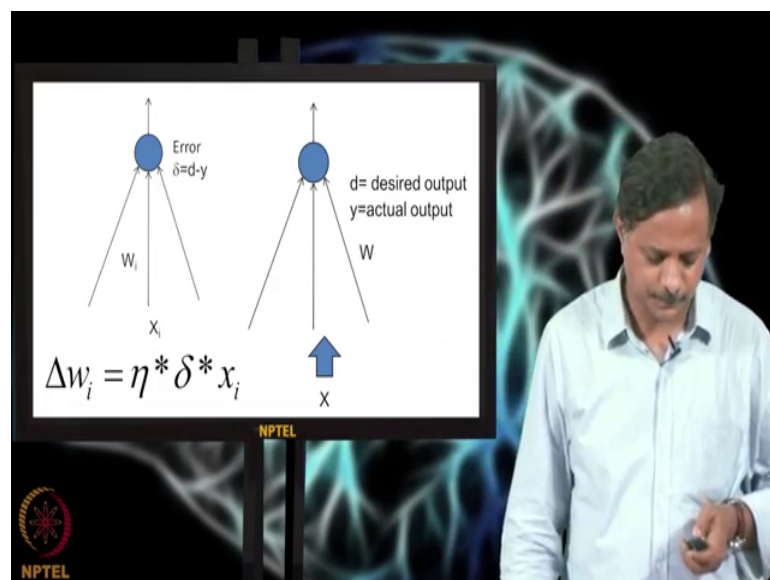


So, you so, in that box in this block diagram in the center you see a box which is which has a neuron, a McCulloch and Pitts neuron. Actual network of them and this network receives an input factor X and it produce an output factor Y and you compare the output

factor to some desired factor or the which is called the target output and look at the difference between the desired output or the target output and the actual output.

If both are the same then so, the 5 network is doing well. If both are not the same, then there is an error. You feed the error back into the network and then there are rules by which you can adjust all the weights slightly. So that next time around when you give the same input. The output that you get from the network which is Y will be slightly closer to the target output. So, you keep doing this alternate will be over and over again or lots of input output patterns right and after lot of such pros you know such train at the network will most likely converge to correct solution, so that it learnt a map all the inputs X onto all the outputs Y.

(Refer Slide Time: 14:42)



So, how does it work? So, you imagine a neuron which takes input factor X and there all these weights that associated with the input connections.

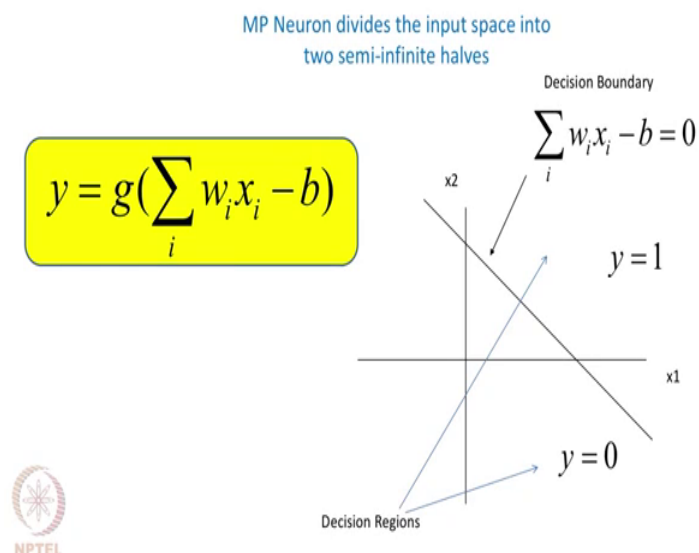
And when you calculate the output y using the formula that you have seen little while ago, you get output y and you can you also have the desired output d you compare the desired output d with y and you get the error delta which is equal to d minus y. And once you know the delta, you can update all the weight W is in the formula given in the bottom of this light. So, delta W i is equal to eta which is like proportionality constant which is typically a small positive number times delta the error times a input x i. So, for every input factor x you represent the input factor get the output calculator at the output

and then use the error which is available at the top and multiplied with the input which is available at the bottom of the connect right and then multiply this product with a with a proportionality constant eta and that that tells you by how much you have to update this weight.

So, if we interpret this biologically, what you are doing essentially is, if you take a synapse. On the presynaptic side of the synapse, you have some information which is the input that is x_i on the post synaptic side your information delta which is the error. And so, at the synapse you imagine that the synapse is taking information from it is postsynaptic side and features there it is taking information from the presynaptic side and fetches input value x_i , somehow combining these two and faring out how to change it is own strength at the synaptic strength. So, that is biologically in a very general sense it is feasible, but synapse is do not exactly behave like this has some issues we will be able to come to it later on the course ok.

So, what kind of what does network like this do? You know if it is a single neuron what is exactly is it doing?

(Refer Slide Time: 16:38)



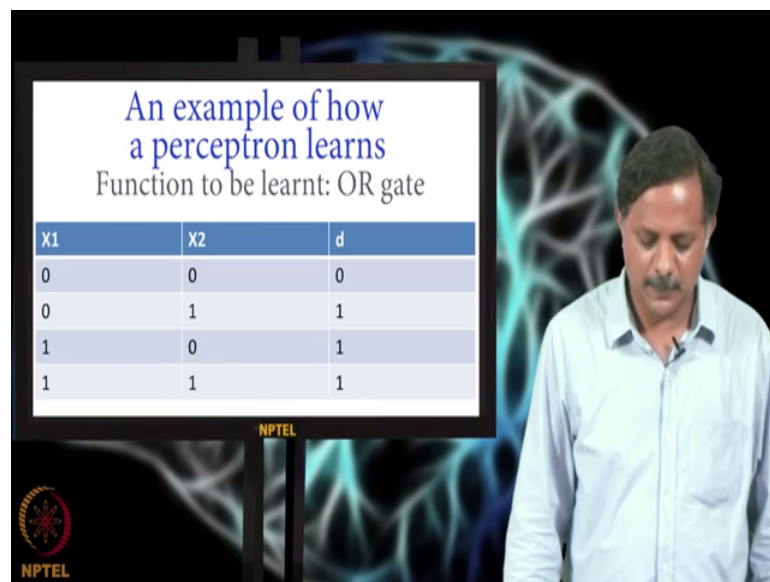
So, consider the formula again y is equal to g of $\sum w_i x_i - b$. So, you see that g is threshold function or the step function. So, therefore, for some values of x , y is equal to 1 and for other values of x , y is equal to 0. And there are there are points where the or the output is a kind of somewhere between 1 and 0. Actually it is a step function that

cannot take any value between 1 and 0, but so, thing is when the argument is positive y is 1 and when the argument is negative y is 0. So, in the argument that is $\sum w_i x_i - b$, if it is 0 right that is why when you are on the border between these two regions.

So, the regions where output y is equal to 1 and output y is equal to 0 are called the Decision Regions because it is as of the neuron is making a decision about the input. So, whether does input belong to one class where output is 1 or does it belong to other class where output is 0, that is decision it is making. Therefore, the 2 regions are called Decision Regions and the boundary between 2 regions which is a straight line in this case is called the Decision Boundary. So, we will use this terms later on when we describe more complex problems.

So, let us see how a perceptron can learn. So, in earlier we showed how you can just hand code the weights of a McCulloch and Pitts neuron model to make it behave like an OR gate.

(Refer Slide Time: 18:00)



An example of how a perceptron learns

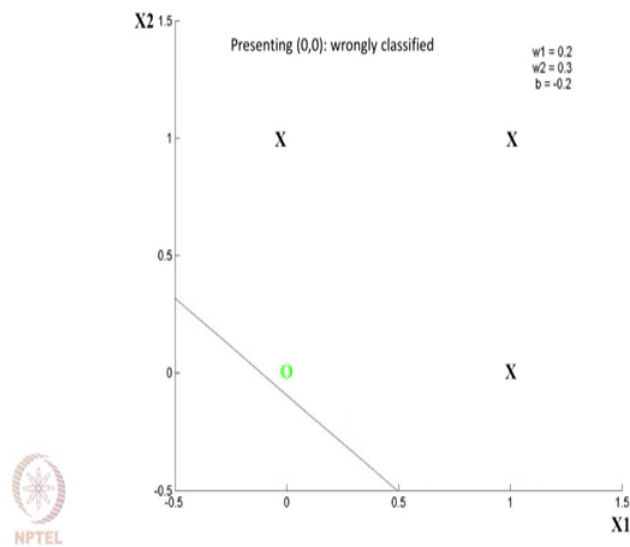
Function to be learnt: OR gate

X1	X2	d
0	0	0
0	1	1
1	0	1
1	1	1

NPTEL

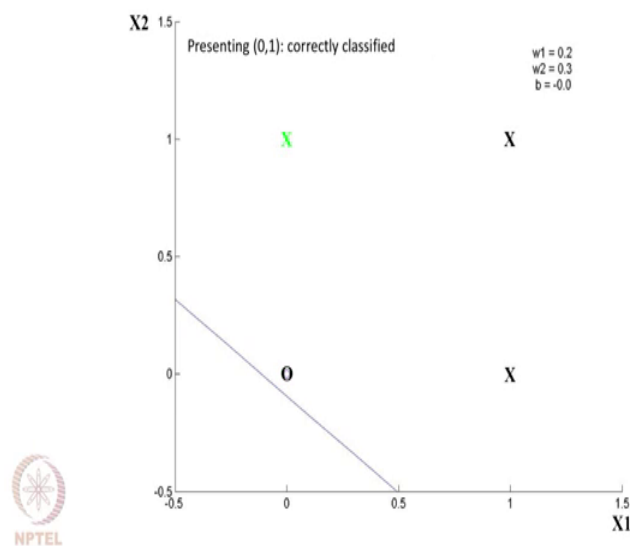
So, now, we will show how you can train the neuron to a on the on the data of the OR gate to make it learn the OR gate function. So, we here the OR gate truth tablet is given here in the table on the slide. So, let us present this data.

(Refer Slide Time: 18:14)



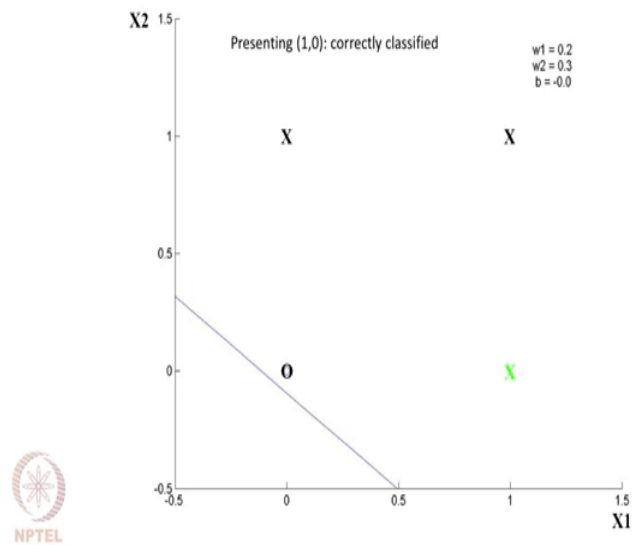
So, I present 0, 0 with the first pattern 0, 0 to the neuron and output should be actually is 0, but it is actually or giving output of 1. So, that is wrongly classified and then you present 0, 1.

(Refer Slide Time: 18:28)



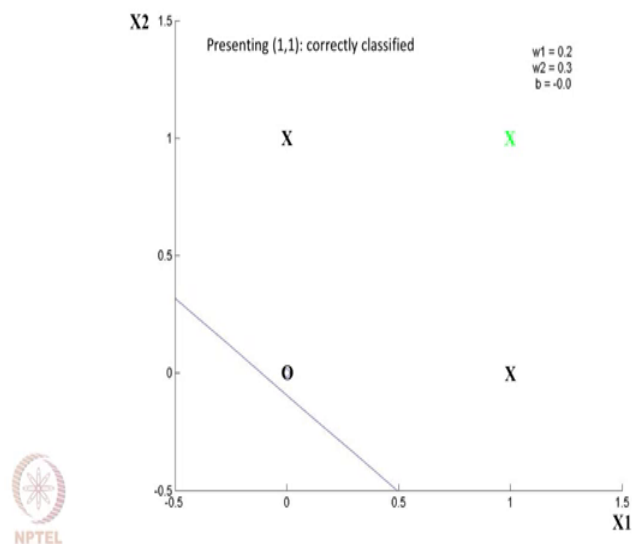
So, the data quantity of presenting is shown on the is shown in color green in this picture itself it is very small, hope you will be able to see it.

(Refer Slide Time: 18:38)



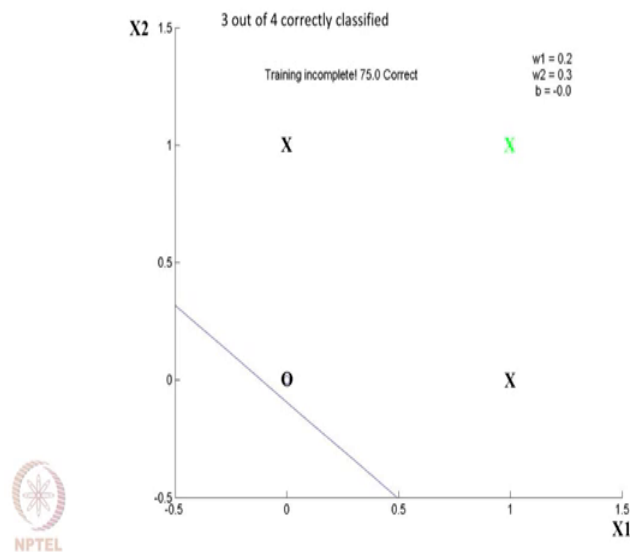
So, in this case the patterns correctly classified. So, there is no change whenever there is correct classification does not change.

(Refer Slide Time: 18:46)



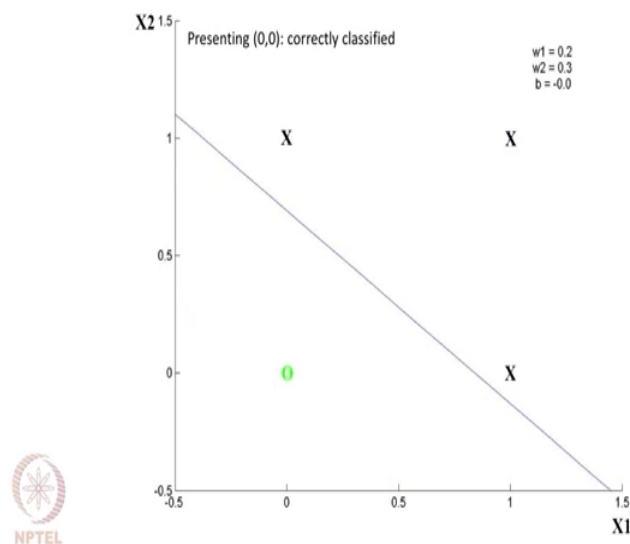
Then you present 1, 1 which is also correctly classified. So, there is no change ok.

(Refer Slide Time: 18:50)



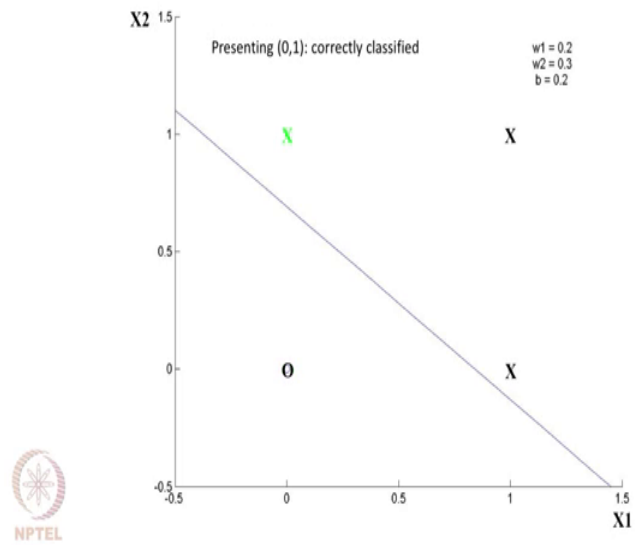
So, out of 4 patterns, the neuron is able to classify 3 patterns correctly ok. So, it you have 75 percent accuracy. So, you have to do it over again until you get 100 percent accuracy.

(Refer Slide Time: 19:02)



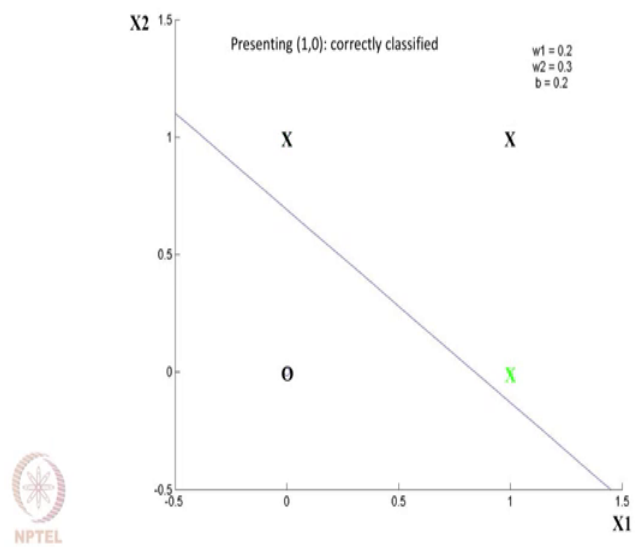
So, present 0, 0 again, this time they line the decision boundary has moved because of learning, the weights have changed a little bit and decision boundary has moved. So, now, the response for 0, 0 is also correct.

(Refer Slide Time: 19:14)



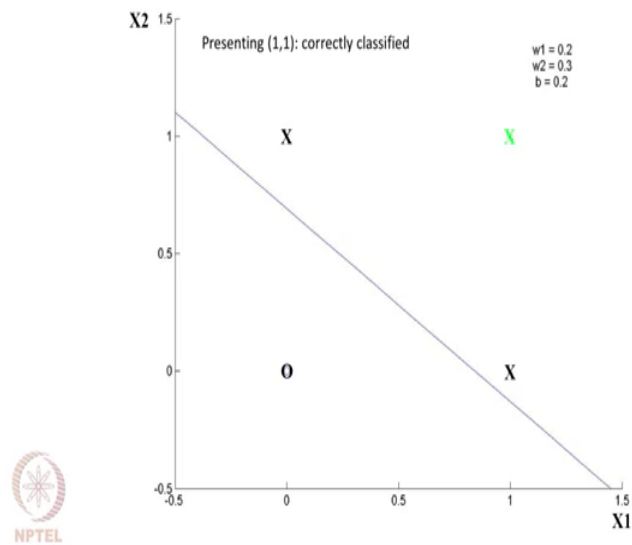
And response for response for the other pattern 0, 1 it is correct, 1, 0 is also correct.

(Refer Slide Time: 09:18)



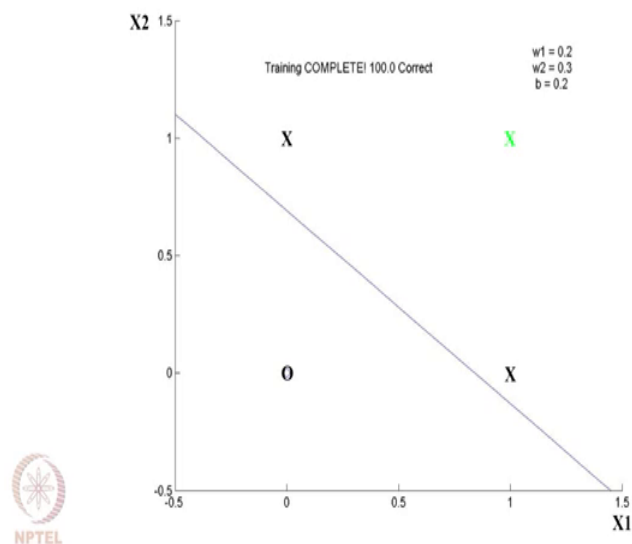
1, 1 is also correct.

(Refer Slide Time: 19:19)



Therefore the neuron has learnt to produce correct responses for all the 4 patterns. So, it is now learned the logic gate OR gate ok.

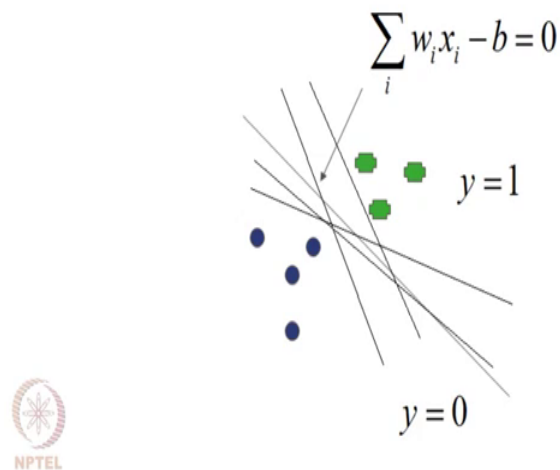
(Refer Slide Time: 19:21)



So, you have 100 percent accuracy. So, thing is what the neuron is doing is it is time to classify 2 sets of points right.

(Refer Slide Time: 19:38)

When the training data is linearly separable, there can be an infinite number of solutions.



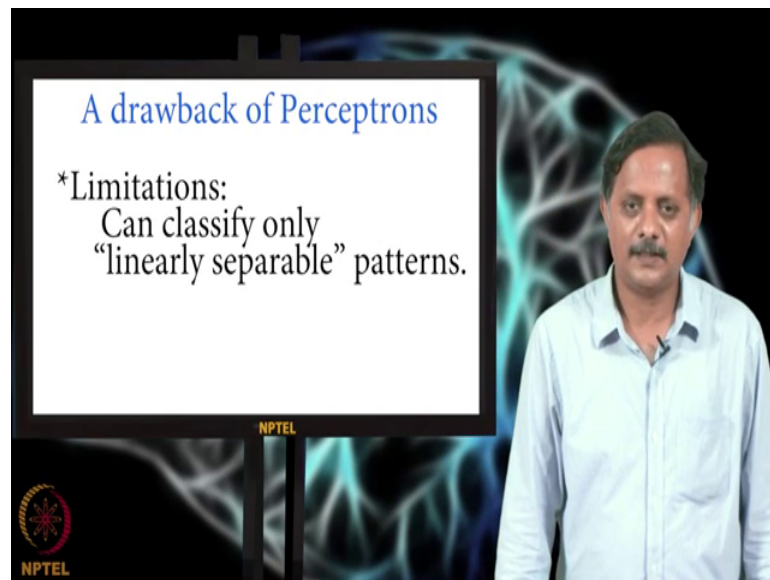
So, in this simple schematic you have.

So, in this simple schematic, you have the green points and the blue blue dots right both represent 2 classes. And the neuron is trying to separate these 2 classes. Identify that it is one is class 1 and one is class 2 the other is class 2 and by drawing a line that separates these 2 classes.

So, in this example we are looking at 2 dimensional data. So, we are ta we are trying to separate the classes single line, but in gena in general case where the input is high dimensional, you separate 2 cases using a ha hyper you know hyper plane right in higher dimensions.

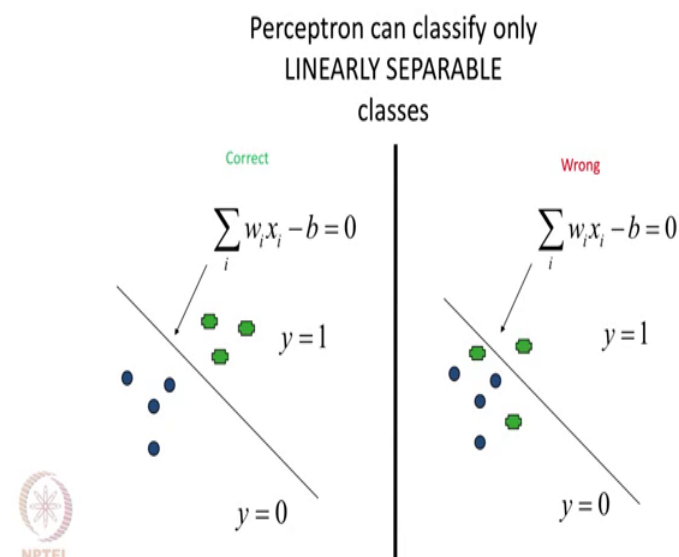
So, and general there can be infinite number of such hyper planes or lines that can separate 2 classes which is what you can say in this picture, but one problem with perceptron is that they can only classify certain special kinds of patterns called linearly separable pattern.

(Refer Slide Time: 20:24)



So, what is that? Let us look at the simple semantic.

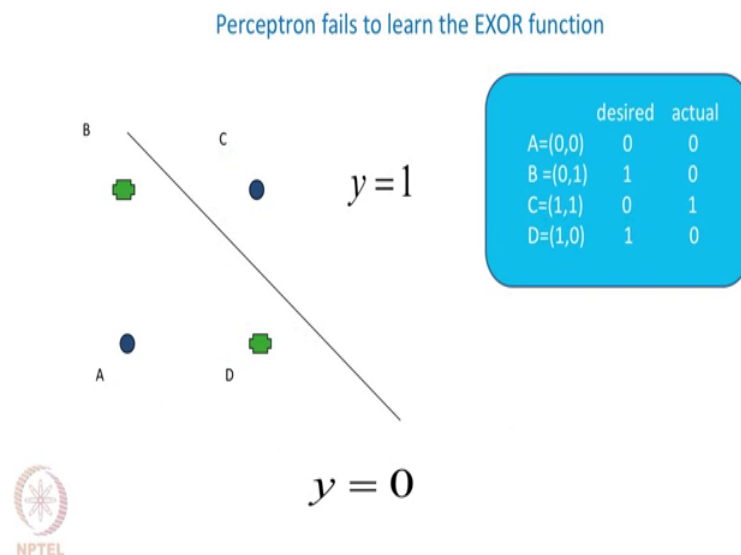
(Refer Slide Time: 20:31)



So, on the left you see 2 2 classes. You have the green dots and the blue dots. You can separate these 2 classes using a straight line, but if you look at data on the right side, again you have green dots and blue dots, but no matter how you draw the line you cannot separate these 2 classes right. You are always miss class where some patterns.

So, patterns which cannot be separated or classes which cannot be separated by a straight line or hyper plane are collinearly non-separable classes. And it turns out that a perceptron cannot learn those kinds of classes. They are too complicated for a perceptron.

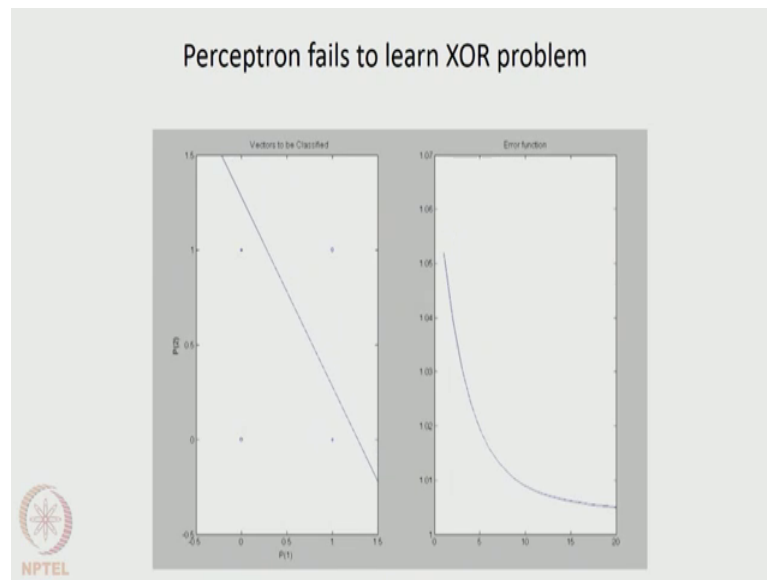
(Refer Slide Time: 21:08)



So, if you take a slightly more complicated logic gate like the EXOR gate whose truth table is given here in the top of the slide right. So, for input 0, 0 you have output is 0 and 0, 1 it is 1 and 1, 1 is 0 and 1, 0 is 1. So, this is depicted in the figure in the bottom the 2 opposite diagonal points. So, A and C belong to the same class and similarly B and D belong to the same class.

Now, no matter how you draw the line that you would not be able to separate the green dot from the blue dots. So, that you can verify right in this little in this video if you put together right.

(Refer Slide Time: 21:48)



On the left side, you are seeing how the decision surfaces decision boundary is moving and on the right side you see the error. So, error should ideally go to 0. If the neuron is able to learn all the all the right responses correctly, error will go to 0. But in this case even after lot of training the error does not go to 0 and stopping at 1. So, it is making a mistake on you know one data point and it it would not be able to recover from that it would not be able to learn that perfectly.

So, the perceptron is interesting model. It can learn supposed to the traditional McCulloch and Pitts approach to modeling brain. So, that is what is nice about a perceptron, but its drawback is it can only learn very simple kinds of classes, classes which can be separated which are linearly separable.

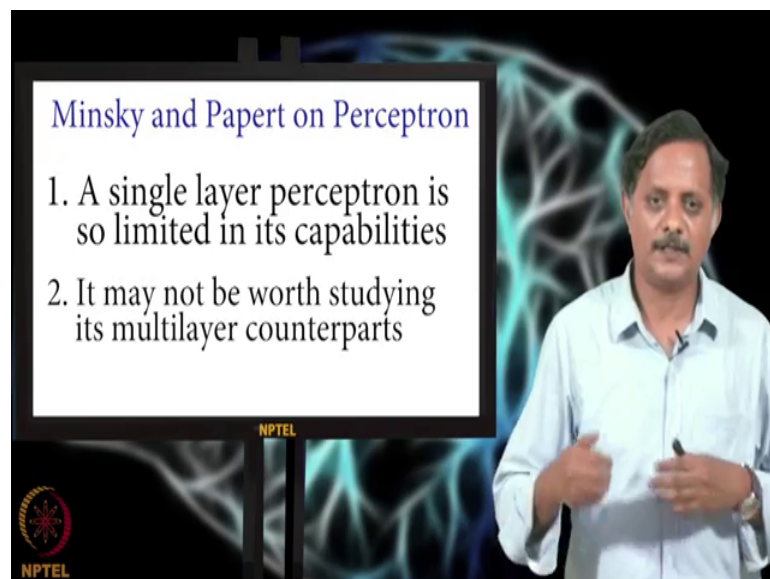
So, therefore, certain others have you know long stay a strong criticism against perceptron.

(Refer Slide Time: 22:45)



So, particularly Minsky and Papert who come mostly from artificial intelligence domain, which is like a rival domain you know for this kinds of neural network models, they launched a very strong criticism saying that the single perceptron limited in its capabilities.

(Refer Slide Time: 22:57)



So, therefore, which is that is true, but then the next statement is not quite correct because they said it worth studying it is multilayer counterparts. So, usually that actually

is a slightly counter initiative because. So, normally when a network which has 2 layers shown to be not very powerful.

The network with multiple layers or more layers is likely to be more powerful because it is bigger, it is more complex. The conclusion that it is the conclusion that Minsky and Papert have drawn is totally contrary that it kind of intuitive understanding.

(Refer Slide Time: 23:34)

Minsky and Papert on Perceptron

“The perceptron has shown itself worthy of study despite (and even because of !) its severe limitations. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many - layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension to multilayer systems is sterile.”

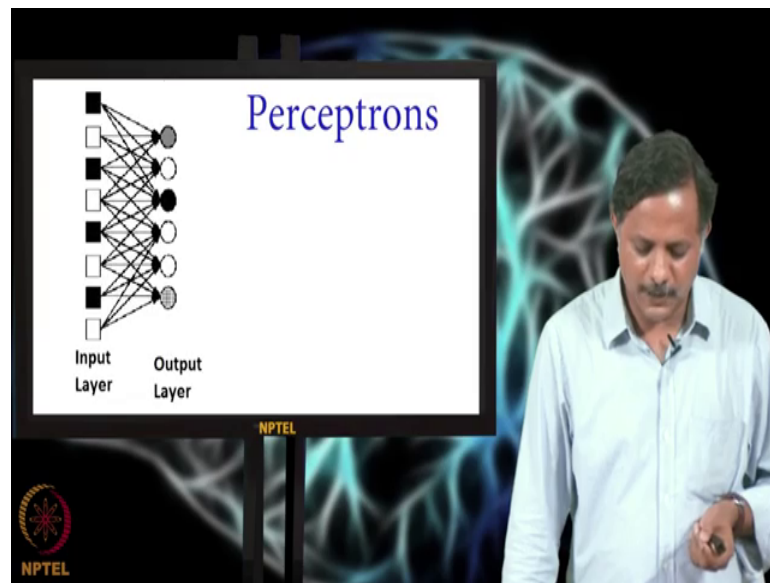


So, in their own words what they have said about perceptron is as follows. “The perceptron has shown itself worthy of study despite or even because of its severe limitations. It has many features to attract attention: it is linearity; it is intuitive learning theorem; it is clear paradigmatic simplicity has a kind of parallel competition.

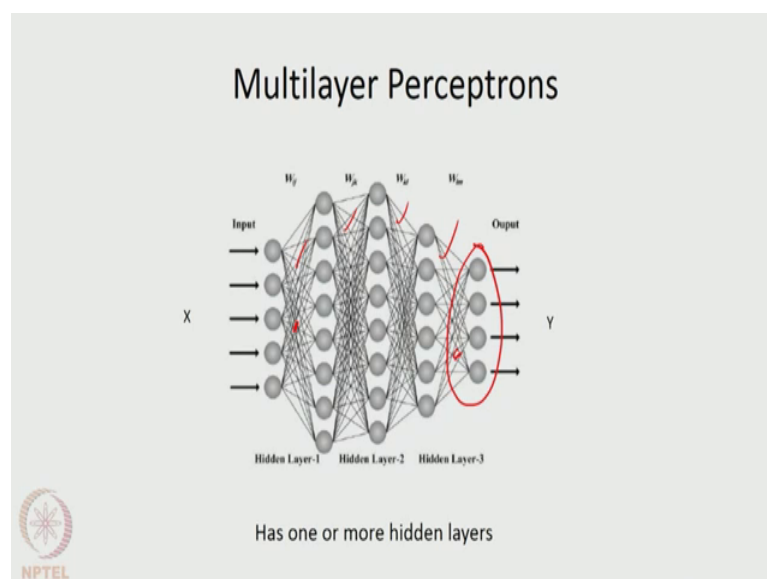
There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider to be an important research problem to elucidate or to reject or intuitive judgment that the extension to multilayer systems is sterile.” Basically they have expressed the prejudices that it is futile to study multilayer versions of a perceptron ok.

In spite of that kind of warning or a caution from Minsky and Papert where people are going just went ahead and studied multilayer versions of a perceptron and actually found that the multilayer versions of perceptron actually lot more powerful and they are free from the some of the limitations of a perceptron.

(Refer Slide Time: 24:31)



(Refer Slide Time: 24:39)

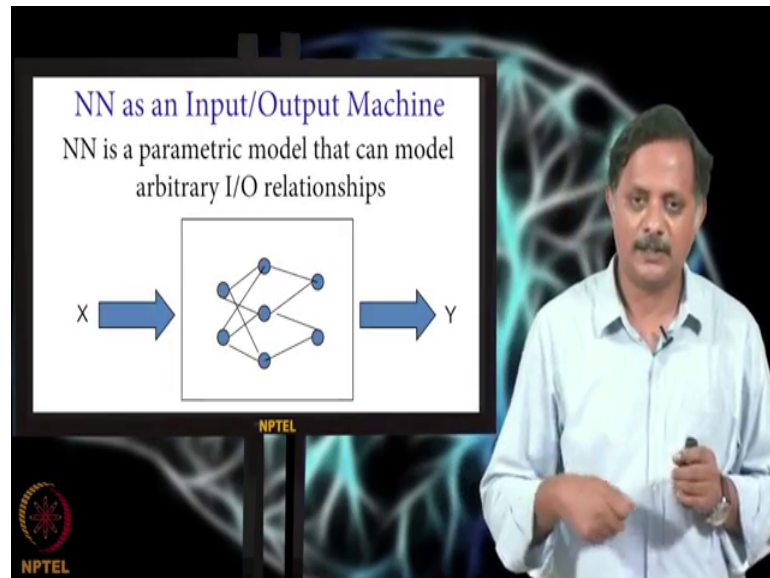


So, if you look at perceptrons have only 2 layers, there is input layer and the output layer and there are no layers in between whereas, in case of Multilayer Perceptrons, they can have any number of layers. So, here also you have input layer right and the output layer. And between the two, you can have any number of intermediate layers which are called hidden layers.

So, in this figure you can see 3 hidden layers right and. So, with the with this large networks, people have found that we can solve lot of problems which you could not have

solved with a perceptron. And in case of multilayer perceptron, the way you think about them is has input/output machines right.

(Refer Slide Time: 25:17)

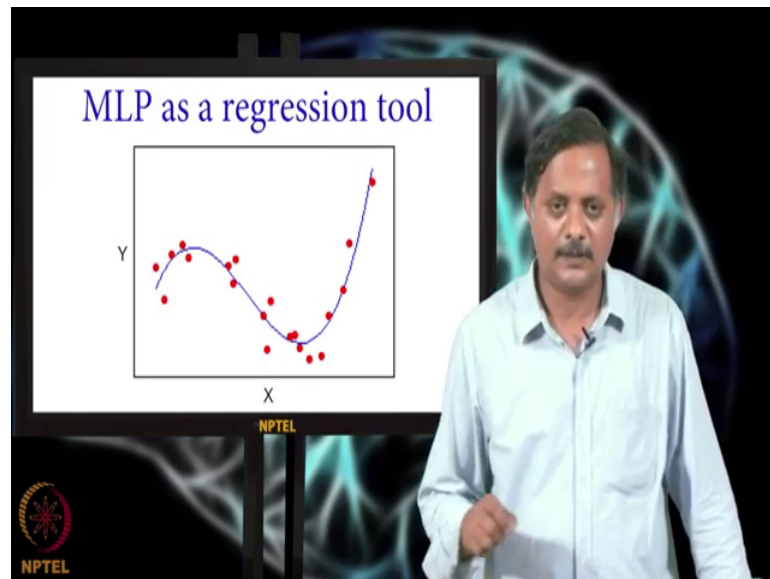


They take the input factor X and produce an output factor Y right and they and the as you train the network and as adjust the weights of the network, the network will learn to map arbitrary inputs and arbitrary outputs. So, they can learn very complex input/output mapping functions.

So, therefore, since they are able to map input onto outputs, you can think of them as some kinds of regression tools.

So, regression or you know more familiarly called curve fitting, that you have some data and you fit a curve to the data.

(Refer Slide Time: 25:44)



So, in this case I have some data which we are able to plot on a plane. So, there is x axis and y axis and you see a bunch of points; bunch of X, Y points shown in this graph. And you fit them with a curve and so, you want to fit a curve which gives you least error at all this data.

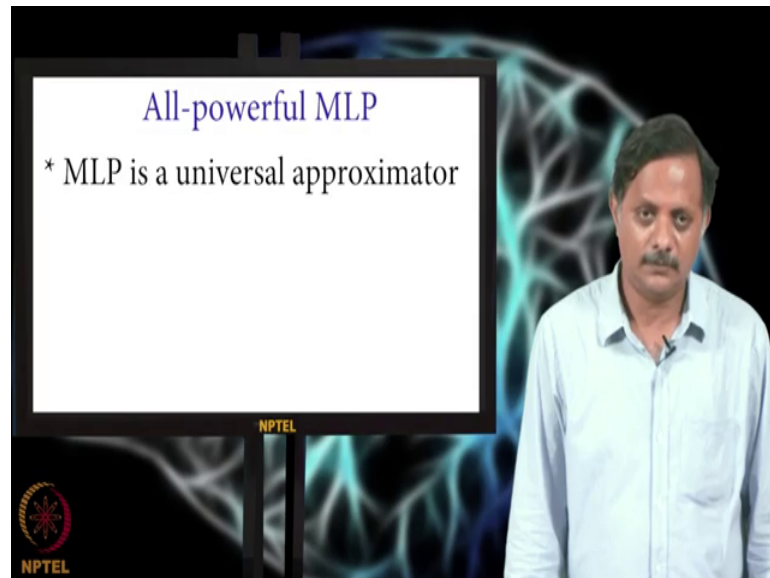
(Refer Slide Time: 26:07)

The figure shows a presentation slide titled "MLP as a regression tool". The slide contains two bullet points: "* Training patterns are the data points" and "* MLP is the Regression Function". The presenter, a man with a mustache wearing a light blue shirt, is standing on the right side of the slide, gesturing with his hands. The background of the slide features a stylized, glowing blue neural network structure. The NPTEL logo is visible in the bottom left corner of the slide.

So, now the training patterns that we use in case of an MLP, are like the data point that use in this graph and the curve which you are fitting into this data is like the MLP itself. So, the input/output function of the MLP is like this curve that you are that you are seeing here. So, basically in curve fitting that is your you have only one input variable X which you are mapping on to a output variable Y.

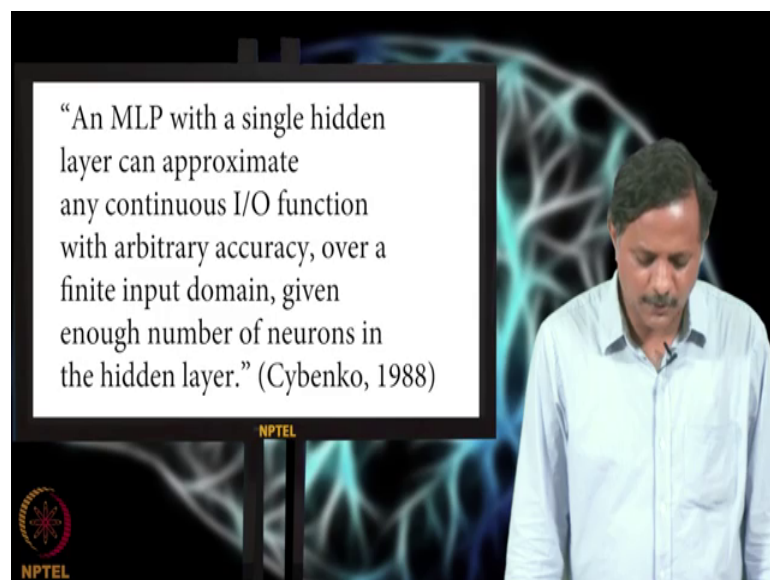
Whereas in case of an MLP, you can have any number of input variables; because X is a vector and which is mapped onto any number of output variables which is a vector Y .

(Refer Slide Time: 26:36)



And it turned out that MLP is free from the kind of limitations that we have seen in case of perceptron MLP is what is called universal approximator.

(Refer Slide Time: 26:47)



Well, what does that mean? It has been shown that an MLP with a single hidden layer can approximate any continuous input/output function with arbitrary accuracy, over a finite input domain given number of given enough number of input neurons in the hidden

layer. So, this was shown mathematically and there is a theorem which you know proves this. So, to illustrate that I mean it this is a very interesting and a powerful theorem in units for literature.

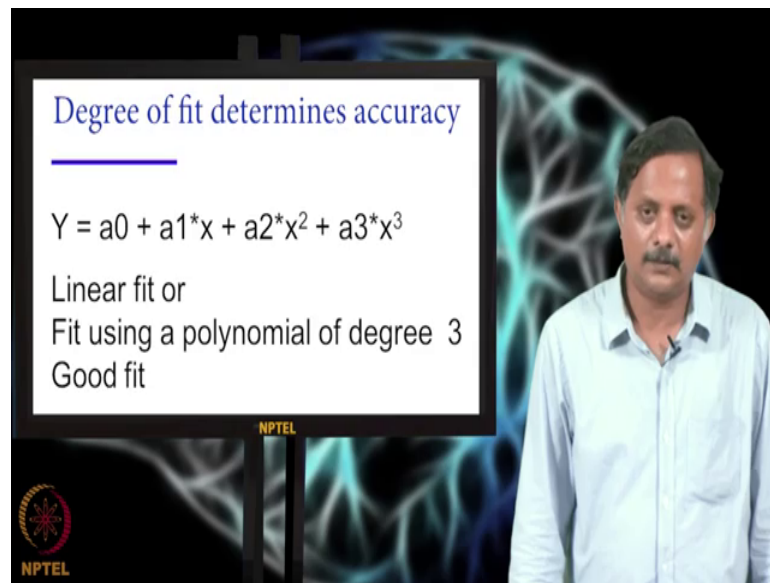
But, we know these kinds of results you know in our simple you know did in regression literature. So, for example, look at this example. So, we have we have on this data which we have to trying to fit with a curve you know just a moment ago.

(Refer Slide Time: 27:34)

The slide is titled "Degree of fit determines accuracy" in blue text. Below the title, the equation $Y = a_0 + a_1 \cdot x$ is displayed. Underneath the equation, the text "Linear fit or" is followed by "Fit using a polynomial of degree 1" and "Bad fit". The slide features a green horizontal line under the title. In the bottom left corner, there is a circular logo with a star and the text "NPTEL". The background of the slide shows a man in a light blue shirt standing in front of a dark background with a glowing blue network pattern. The NPTEL logo is also visible in the bottom center of the slide.

So, let us assume, we try to fit a straight line to this data and which you can see this green line is a straight line which we are trying to fit to the data and it is a very bad fit right so; that means, if you fit the formula Y is equal to $a_0 + a_1 x$ at it is a linear fit; that means, you are fitting it with a polynomial of degree 1 right, you get that green line which is a very bad fit. But on the other hand, if you fit a cubic function to this data that is Y is equal to $a_0 + a_1 x + a_2 x^2 + a_3 x^3$, you get this blue curve right which is a which is a much better fit and here the polynomials of degree 3.

(Refer Slide Time: 27:54)



Degree of fit determines accuracy

$$Y = a_0 + a_1x + a_2x^2 + a_3x^3$$

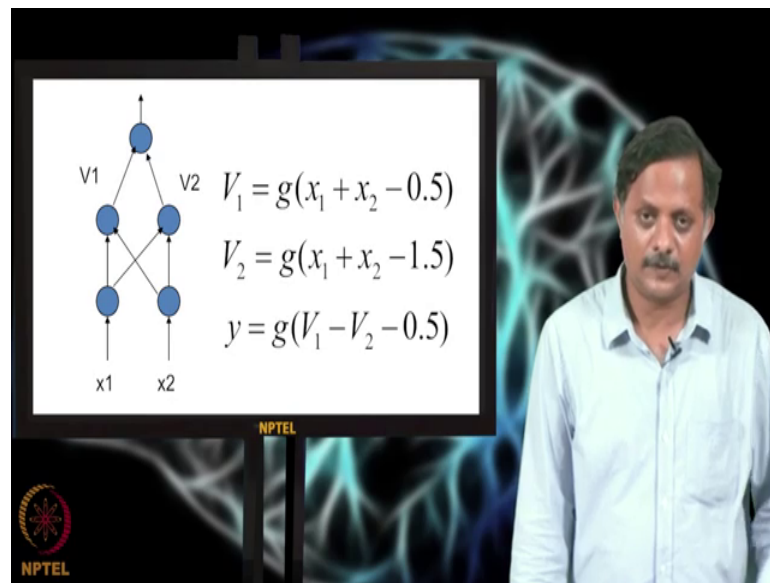
Linear fit or
Fit using a polynomial of degree 3
Good fit

NPTEL

So, basically what you are saying is a simple example is, as we increase the degree of the polynomial with which you are feeding the data, that if it gets better and better. So, what we were seeing in case of neural network is something very similar. So, as you increase the number of hidden nodes right and if it gets better and better. So, that is what the essence of this theorem is what this theorem says is, a network with a single hidden layer is good enough to learn a wide range of real world functions right. And if you have enough number of hidden neurons in the hidden layer, then you will be able to learn any function with arbitrary accuracy.

So, let us take a problem which was not solved by a perceptron and we will show using some simple calculations that an MLP can solve this problem. So, let us take the example.

(Refer Slide Time: 29:00)

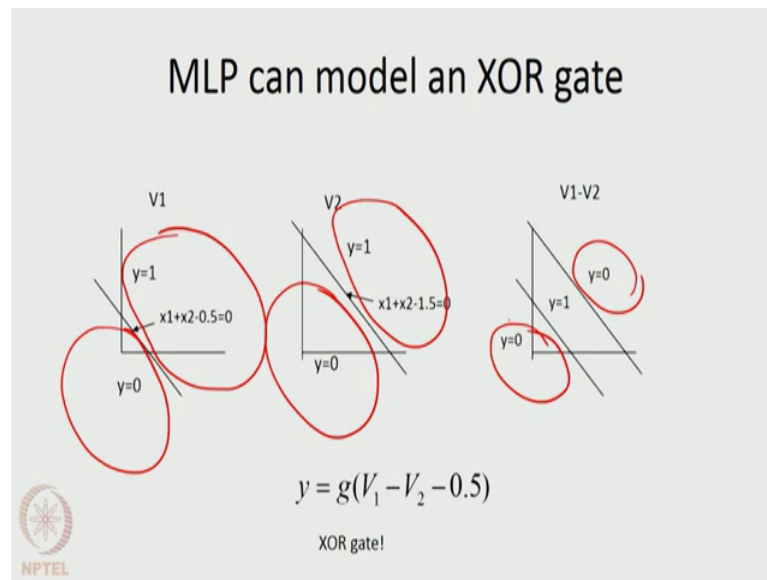


So so so, we will take MLP which has a single hidden layer and that hidden layer has 2 neurons. So, the input to the network is two variables x_1 and x_2 and there are two hidden neurons and their outputs are v_1 and v_2 and finally, the outputs the v_1 and v_2 neurons project to a single output neuron and whose output is y .

So, therefore, the input/output relationships of all these neurons are given by this formulae. So, V_1 is given as g of x_1 plus x_2 minus 0.5. So, that is like an OR gate and V_2 is given as g of x_1 plus x_2 minus 1.5. So, that is like an AND gate, which you have seen before and y is equal to g of V_1 one minus v_2 minus 0.5.

So, what is happening here is, so, V_1 is given by g of x_1 plus to x_2 minus 0.5.

(Refer Slide Time: 29:50)



So, therefore if you see in this figure, so, this whole region is where V_1 is equal to 1 and this whole region V_1 is equal to 0 and in this case this whole region V_1 is 1, here V_1 is V_2 is 0 and here V_2 is 1.

So, if you take V_1 minus V_2 , you get a kind of a band where otherwise 1 this is the center inside this band and y is 0 outside on either side of the band ok. So, therefore, y is equal to g of V_1 minus V_2 minus 0.5 alright approximate.

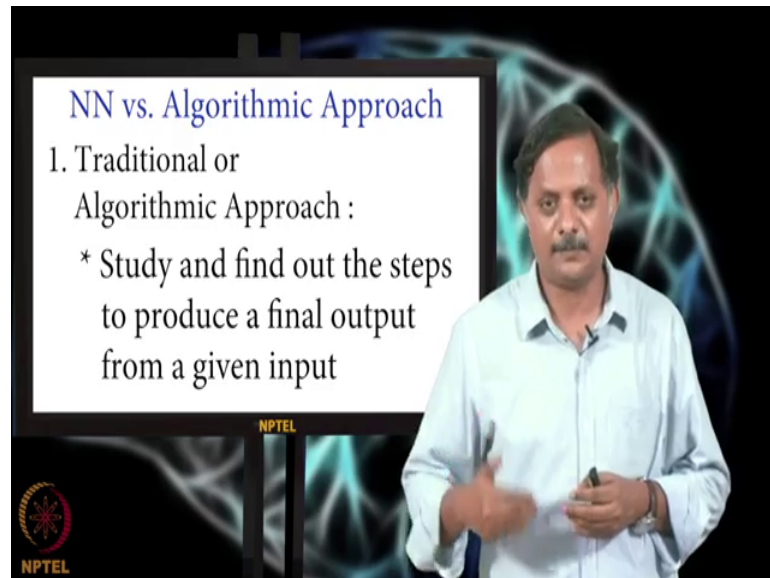
So, the EXOR gate it learns EXOR gate. But what you have done just now is a kind of a simple hand calculation, that just we wanted to show that a network which has a hidden layer can do something that a perceptron cannot do. But you cannot do the same thing with you know large scale real world problems.

So, therefore, we will have to come up with a learning algorithm a learning scheme by which you can paint the weights automatically. So, the network will learn the relationship to input and output automatically without does calculating the weights by hand.

So, we will switch back to the same learning scheme right where you have the network inside this box the NN and then input is present to the present to the network and you get the output and you compare the output with a target output or exit output look at the error and error is set fed back to a network and using the errors you update all the weights

slightly. So, next time around when you give the same input X , you get the output Y which is closer the target output T .

(Refer Slide Time: 31:31)



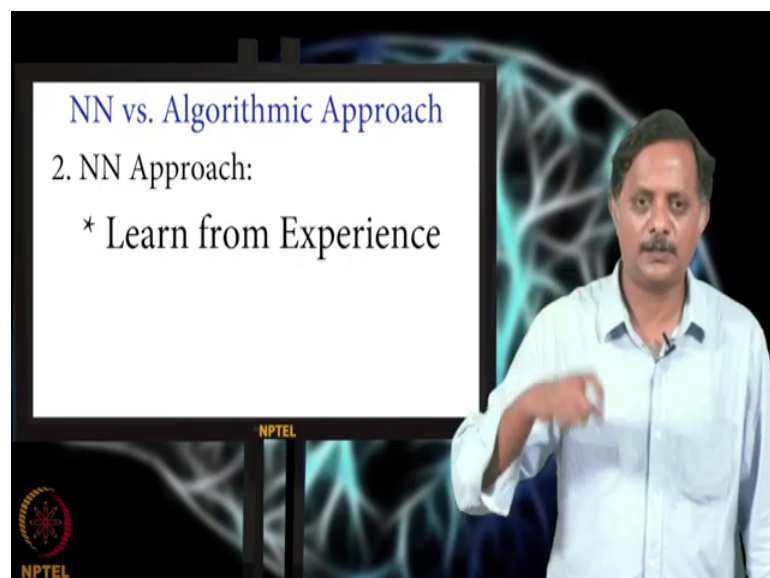
Slide titled "NN vs. Algorithmic Approach". The slide content is as follows:

- 1. Traditional or Algorithmic Approach :
 - * Study and find out the steps to produce a final output from a given input

The slide features a background image of a neural network and the NPTEL logo in the bottom left corner. A presenter is visible on the right side of the slide.

Now, this is a actually a very interesting departure from the traditional way of solving problems. So, traditionally we want to solve a problem right in which consists of transforming certain input quantities X into some output quantities Y and you have to figure out what is the relation between X and Y and come up with a set of rules set of calculation steps and execute those steps to transform X into Y .

(Refer Slide Time: 31:51)



Slide titled "NN vs. Algorithmic Approach". The slide content is as follows:

- 2. NN Approach:
 - * Learn from Experience

The slide features a background image of a neural network and the NPTEL logo in the bottom left corner. A presenter is visible on the right side of the slide.

But with this kind of neural network approach right, you just learn from experience. All that you do, is expose the network to lots of inputs and outputs, these are examples. These are increases some experience and the network will automatically learn to do this map by training.

So, this is also very similar to how we learn right, where you how you teach say somebody by trial and error. So, it just visualize how you tea teach a small child how to read. So, perhaps you will show the child a picture of a letter A and also child to pronounce it as A, if the child says A you know you give some you know pat on the back or give some lollipop or something like that.

While the child says B then that is a wrong response. So, then you give a warning or a little slap or something like that. And you by this kind of error feedback, where the child gradually learns to to read the letters correctly. So, that is that is exactly now an MLP is trained to learn the map.

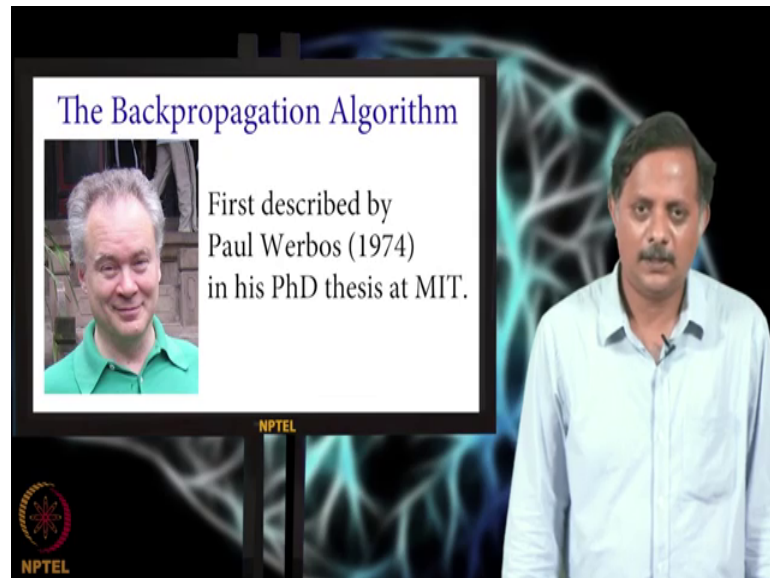
So, the question of training the network; training the MLP right developing an algorithm for training MLP, initially ran into some difficulties because in case of perceptron, if it is very easy how to derive the learning rule because in case of perceptron, you had an error at the output and you had input x_i and by multiplying these two, you are able to update the weights.

But whereas, in case of an MLP, there is a problem; because it had many layers and otherwise available at the output layer, and you have to change the weights everywhere all these stages ok. So, now, you need error throughout the entire network update the weights, how could you use error which is available here to update the weights which are very far from the place where errors are available? Ok. So, this was the challenge and initially for it was not clear how to overcome this challenge. So, this problem is called credit assignment problem or right and, but very soon it was solved by several people. there is a long history of how people have discovered this algorithm over and over many people have discovered this algorithm now again and again without the knowledge and again without knowing the knowledge that others have already discovered it.

Thing is after the severe criticism of Minsky and Papert in the 60s about perceptrons, for a long time nobody bothered about these kinds of network models nobody bothered to study multilayer versions of perceptrons. So, there was a (Refer Time: 34:21) research

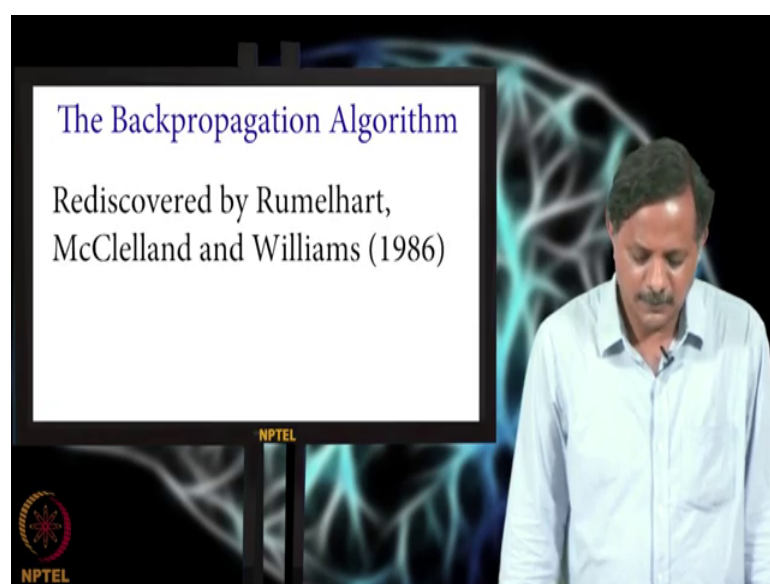
for nearly 2 decades from 60s and only in the 80s you know the research has picked up again.

(Refer Slide Time: 34:32)



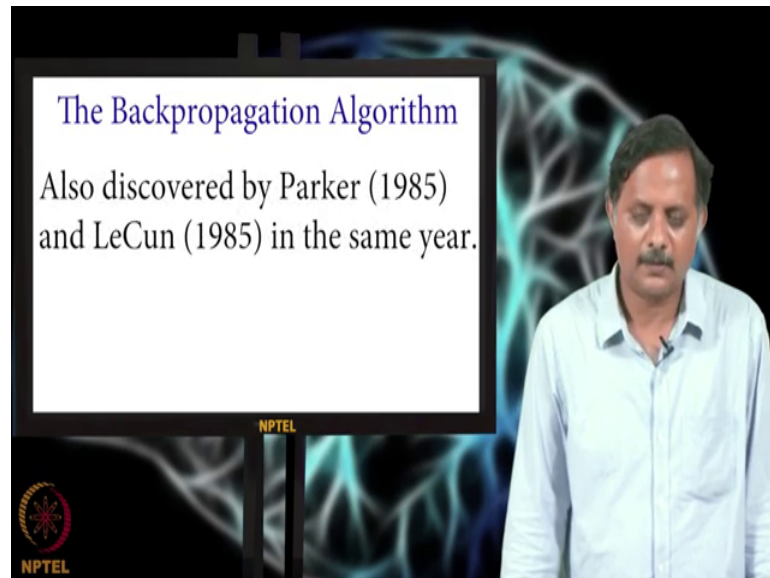
But in the 70s, there was this solitary hero called Paul Werbos who has read thesis at MIT on the learning algorithm for multilayer perceptrons, but nobody paid attention to his work because this area was not popular. It was like a taboo to work on multilayer perceptrons.

(Refer Slide Time: 34:47)



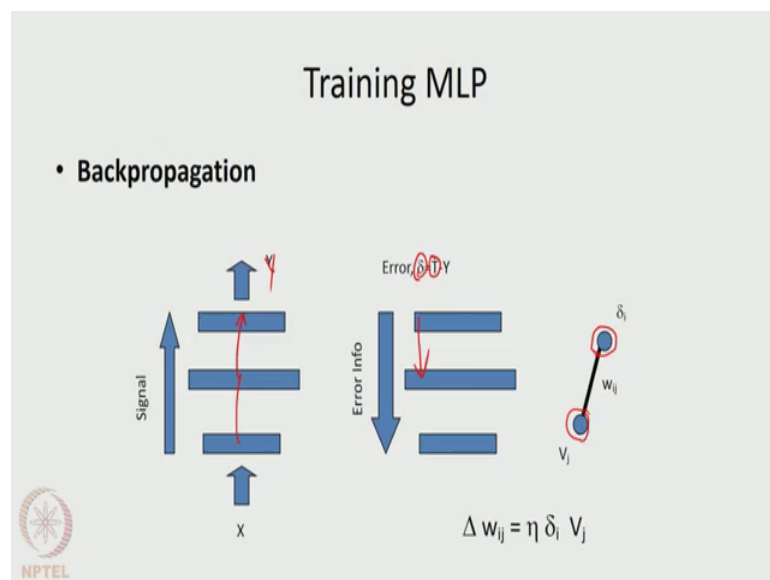
So, later on in the 80s several groups have discovered the same algorithm nearly simultaneously. So, one was Rumelhart, McClelland, Williams in 86.

(Refer Slide Time: 34:55)



Then Parker in 85 and LeCun in 85, they all discovered the same algorithm independently.

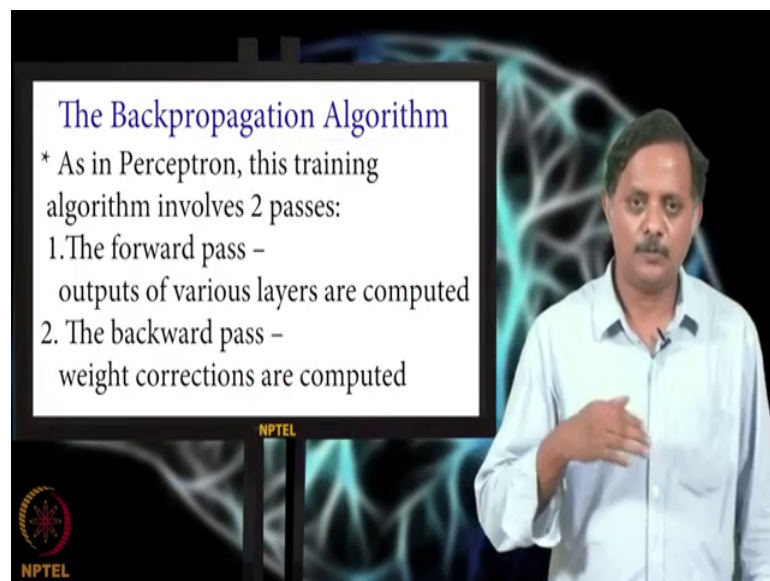
(Refer Slide Time: 35:02)



So, in the multilayer perceptron which is trained by this algorithm there are 2 stages. In the first stage so, the input X is presented to the network to the input layer and then it is made to climb upwards right and from input to the hidden layer a hidden layer to the output layer and so on. And the output layer you get the output Y . Then you compare the output Y with a desired quantity desired output T right and you get the error which is delta that is T minus Y .

Once you have the delta, you back propagate this delta right from output layer to hidden layer. And if you have more hidden layers, you back propagate the error further down, all the way to the first hidden layer. And once you have all the deltas for our all the hidden layers of the network, then you can update all the weights in one shot using like this. So, for any given connection, you take the delta of the neuron at it is post-synaptic side and you take the output of the neuron on it is pre pre-synaptic side and multiply these two and multiply the product with a step size η right, you know you have the update value for the corresponding weight.

(Refer Slide Time: 36:23)

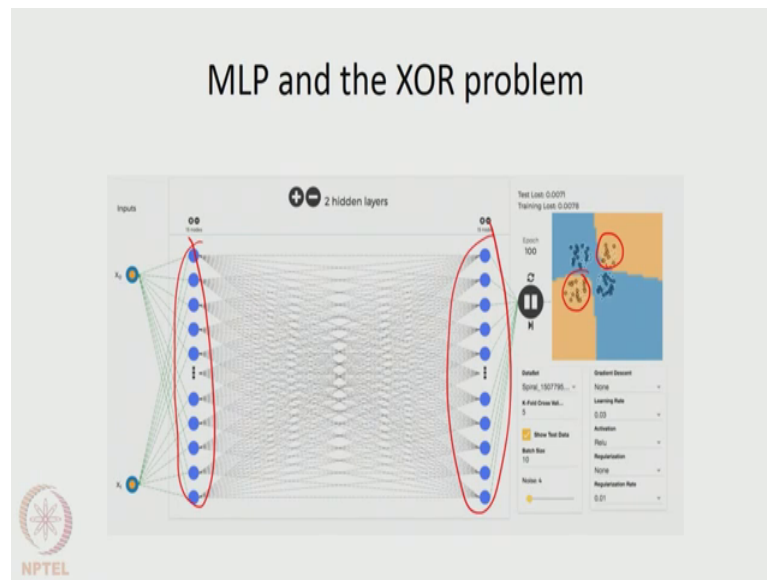


The Backpropagation Algorithm

- * As in Perceptron, this training algorithm involves 2 passes:
 1. The forward pass – outputs of various layers are computed
 2. The backward pass – weight corrections are computed

So, this algorithm is called a back propagation algorithm; because in this algorithm the you have 2 passes. The input first goes all the way to the output in the forward pass and reverse pass or the back propagation pass right the errors are back propagated from output all the way to the first hidden layer. And then you update all the weights in one shot.

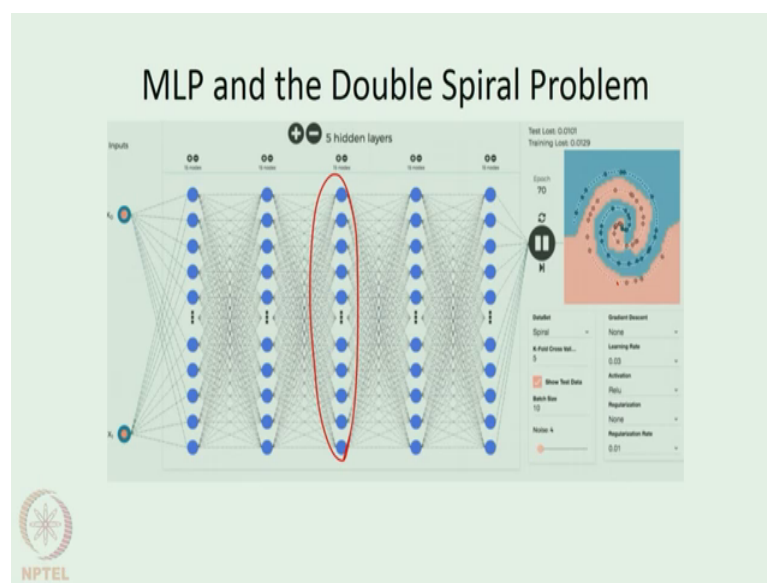
(Refer Slide Time: 36:39)



So, let us see how you can train an MLP on XOR problems, so, what you have done before is we hand calculated all the weights and showed how an MLP can let learn exact problem. So, in this case, we take a huge MLP this has so, 2 hidden layers. So, there is one hidden layer here, which has 15 neurons; another hidden layer here, which has again 15 neurons. And then there is a input and you can see the inputs on the left and that is in the single output Y and you have the exact data is given like this. So, these two yellows belong to one class and the blues belong to the second class.

So if you train this network, let us see how it what happens. So, the reds and yellows that you see in the moving animation on the right side, are the decision regions right. So, the yellow region should fill the yellow dots and the blue region should fill the blue dots and this already happened very quickly; it learns very quickly right. So, network is able to learn this task.

(Refer Slide Time: 37:44)



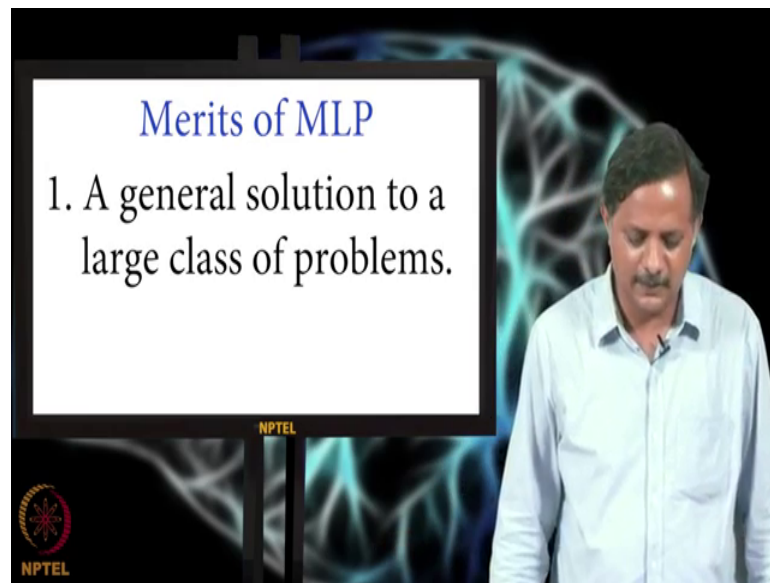
Now, let us look at a more complicated problem. So, in case of XOR, it is more complicated than the you know it is linearly non-separable, but it is still not very complicated. So, take this example where there are 2 classes red and this is called the double spiral problem. The 2 glasses consist of 2 sets of data right. One is this yellow line, the yellow spiral, the other is the blue spiral ok. So, it is very hard to I mean you cannot just draw a line and separate these 2 data sets right. So, it is a very hard problem and let us see how a an MLP can solve this.

So, in this case MLP has only 1 hidden layer and it has just 15 nodes, a 15 neurons. So, let us see how this learns it. So, again you see the 2 decision regions. The yellow region should fill is suppose to fill ultimately; only the yellow points and the blue region should fill only the blue points. But it did not happen you know it is. So, let us erase this ok. So, in this case, now we are looking at much bigger net network because 3 hidden layers each 750 neurons and this is this kind of getting there very quickly. So, the yellow regions only fill the yellow part.

The next, we are looking at 5 hidden layers at each 750 neurons ok. This is learning very quickly. So, you can see the yellow decision regions filling the yellow dots. We almost there already right, it is it is a learnt you know it is learnt the yellow region is filled the yellow dots completely and the blue regions are filled the blue dots completely.

So, you can see that an MLP with multiple layers can learn pretty complicated functions.

(Refer Slide Time: 39:53)



So that is what is nice about an MLP. It is a general solutions and large glass of problems. So, if you want to model for example, something that is happen since happening in the visual system right, you are looking at an image and you want to explain how it passes of image.

So, whether in the brain, we are looking at Visual processing or Auditory processing or processing touch or Somatosensory processing, it is all done by the same set of neurons right of similar neurons whereas, similarly, in case of MLP right, no matter what your problem domain is, is the same kind of a network with same kind of neurons and in a generalized architecture generalized learning rule can be applied to solve the problem. So, that is one thing something that is very appealing about the an MLP.

(Refer Slide Time: 40:35)

Merits of MLP

2. With sufficient number of hidden layer nodes, MLP can approximate arbitrary target functions.

NPTEL

And also we have this theorem which guarantees that given enough number of hidden neurons. The MLP can approximate arbitrarily complex you know target functions right.

(Refer Slide Time: 40:47)

Merits of MLP

3. Backprop applies for arbitrary number of layers, partial connectivity (no loops).

NPTEL

Now, also your back propagation algorithm also applies for arbitrary number of layers and even if you have partial connectivity. The examples that we have seen here involve full connectivity; that means, from every layer to the next layer, all the neurons are connected. Every possible connection exists whereas, even if you leave out some

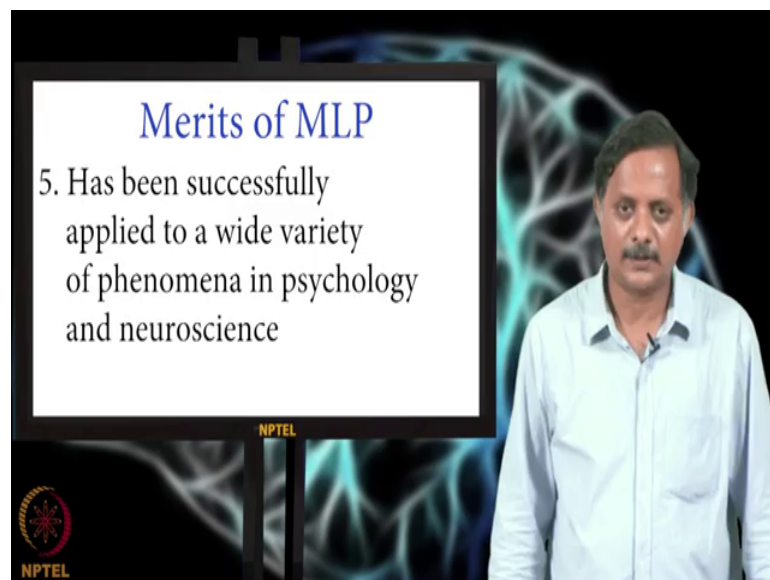
connections right the same algorithm applies, the only condition for the algorithm to work is that there should not be any loops in the network.

(Refer Slide Time: 41:13)



The slide is titled "Merits of MLP" in blue text. Below the title, point 4 is listed: "4. Training is local both in time and space - parallel implementation made easy." The slide is part of a video lecture, with a man in a light blue shirt visible on the right side. The background features a stylized neural network diagram. The NPTEL logo is visible in the bottom left corner.

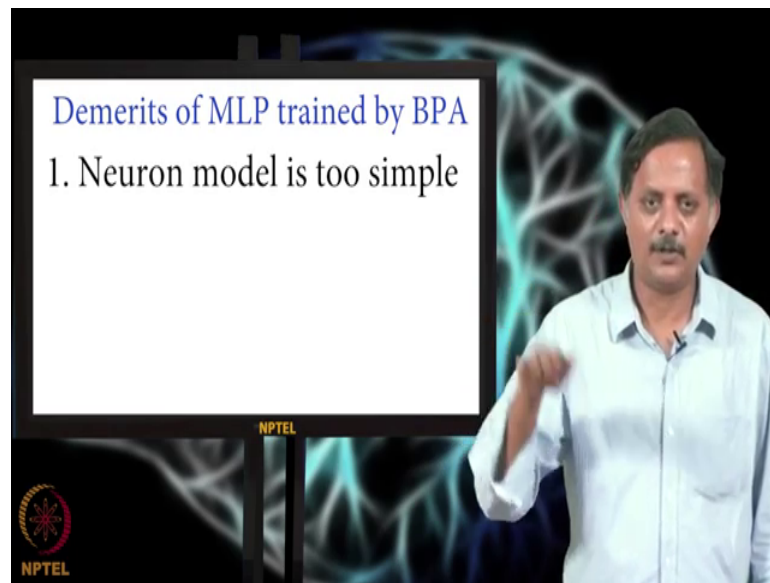
(Refer Slide Time: 41:20)



The slide is titled "Merits of MLP" in blue text. Below the title, point 5 is listed: "5. Has been successfully applied to a wide variety of phenomena in psychology and neuroscience." The slide is part of a video lecture, with a man in a light blue shirt visible on the right side. The background features a stylized neural network diagram. The NPTEL logo is visible in the bottom left corner.

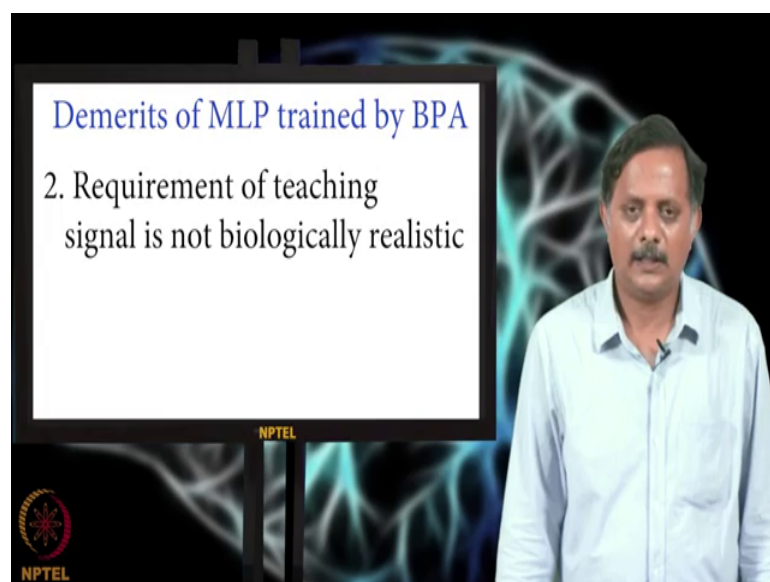
So, and then the network can be trained in parallel and it is been also what is interesting for us right in this course is that the network can be has been applied a wide variety of phenomena in psychology and neuroscience.

(Refer Slide Time: 41:33)



But there is an drawbacks of this network model especially from neuroscience point of view which is at the neuron model is too simple, I mean the real we have we have seen what kind of assumptions we have made right and how far we have moved right from realistic neuron with all the spiking activity and the kind of McCulloch and Pitts neuron are where the output is simply 1 or 0.

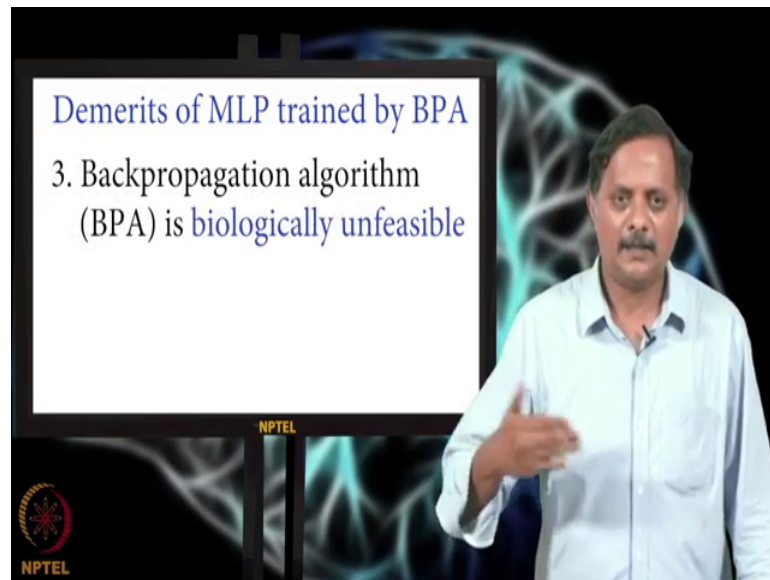
(Refer Slide Time: 41:50)



So, another artificial feature about the MLP is that it requires a teaching signal what is called a signal that is the target vector right. So, we have seen that, if you want to train

the network read the whole target vector T . Whereas, Z is biologically an unrealistic you know you do not have that kind of signal available in the brain to tell various neurons how they should learn.

(Refer Slide Time: 42:16)



Then another aspect of backdrop which is artificial is that it is biologically unreasonable and feasible because. So, the algorithm involves back propagation of errors right downwards towards the input layer. Now, so, we know that synapse is a unidirectional you know signaling structure. So, signal can only go from pre-synaptic side to the post-synaptic side; because the new transmitter is present only on the presynaptic side and it then goes to the pros post synaptic side. Whereas, in back propagation algorithm, you want the signal to be able to go from output to the input and that is not possible.

So there people have tried to justify the back propagation algorithm by invoking certain features of neurobiological like no back propagating action potential things like that, but we refer into those details at this point. But what is interesting is, even this very simplified generalized networks that can explain certain phenomena from psychology and neuroscience.

So, what we will do in the next segment of this lecture is, we will take several phenomena from psychology and neuroscience and show how the networks can be applied to these problems.