Intelligent Systems and Control

Prof. Laxmidhar Behera

Department of Electrical Engineering

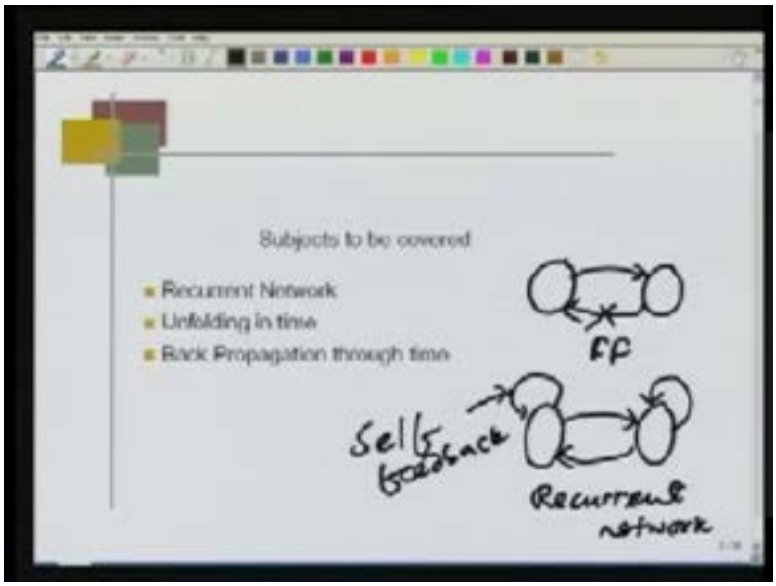
Indian Institute of Technology, Kanpur

Module – 1 Lecture – 9

Recurrent Networks: Back Propagation through Time

This is lecture 9 of module I – neural networks in this course on intelligent control.

(Refer Slide Time: 00:36)

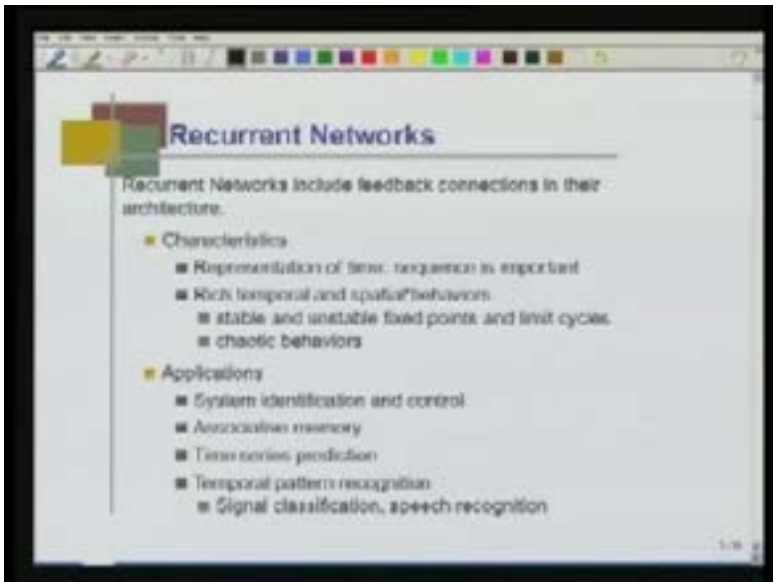


In this course, today, we will cover what is recurrent network, unfolding in time of a recurrent network. What is a recurrent network? A feed-forward network without feedback connection, recurrent network with feedback connection. Unfolding in time, that is a recurrent network, when unfolded in time, can look like a feed-forward network. Then, back propagation through time. This is the learning algorithm for recurrent network. What is a recurrent network?

Earlier, we said that if there is a neuron and there is another neuron, if I connect this neuron to this neuron, then this connection was not allowed in feed-forward network, whereas in a recurrent network, we allow not only the feedback connection but also self-

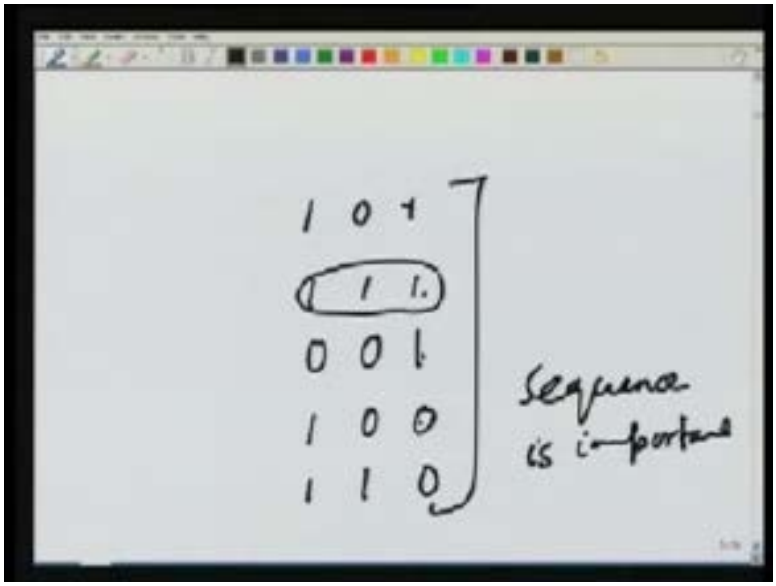
feedback. This is called recurrent network. When we have two neurons in any feed-forward network – multilayer network or radial basis function network, if I connect in a forward direction, the backward direction connection is not allowed, whereas in recurrent network, not only the forward and feedback connections are allowed but also self-feedback. This is known as self-feedback (Refer Slide Time: 02:46). This kind of structure is a recurrent network.

(Refer Slide Time: 02:53)



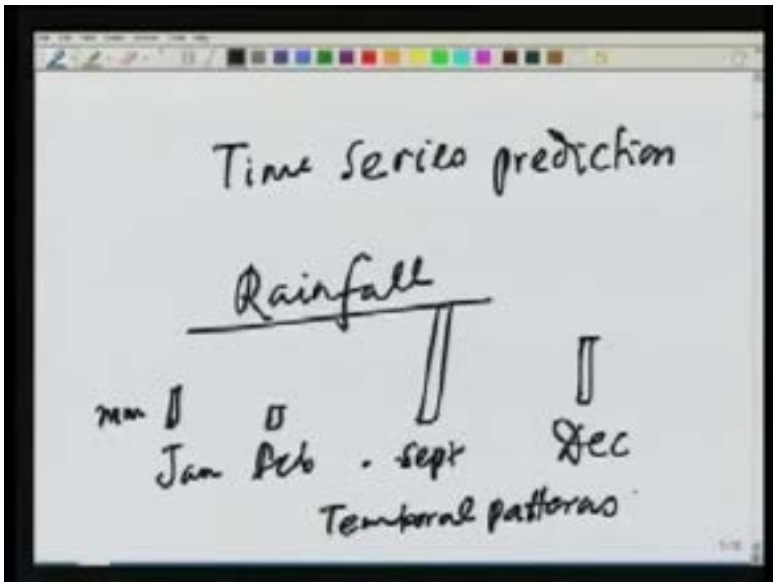
Recurrent network includes feedback connections in their architecture. What are the characteristics of a recurrent network? Representation of time: sequence is important. I would like to explain this particular thing in detail.

(Refer Slide Time: 03:24)



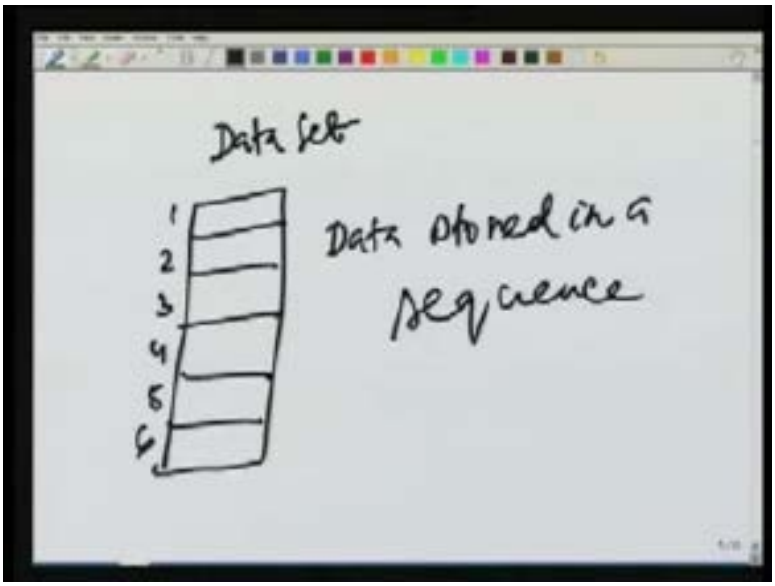
What is the meaning of sequence? Imagine we have five patterns – 1 0 1, 1 1 1, 0 0 1, 1 0 0, and 1 1 0. The meaning of sequence is that this pattern follows this pattern, this pattern follows this pattern (Refer Slide Time: 03:55), this pattern follows this pattern, and this pattern follows this pattern. Imagine there is a display board and I want to display this pattern first, this pattern second, this pattern third, fourth, fifth and rotate it clockwise. In that sense, the sequence is important here. In this case, sequence is important.

(Refer Slide Time: 04:48)



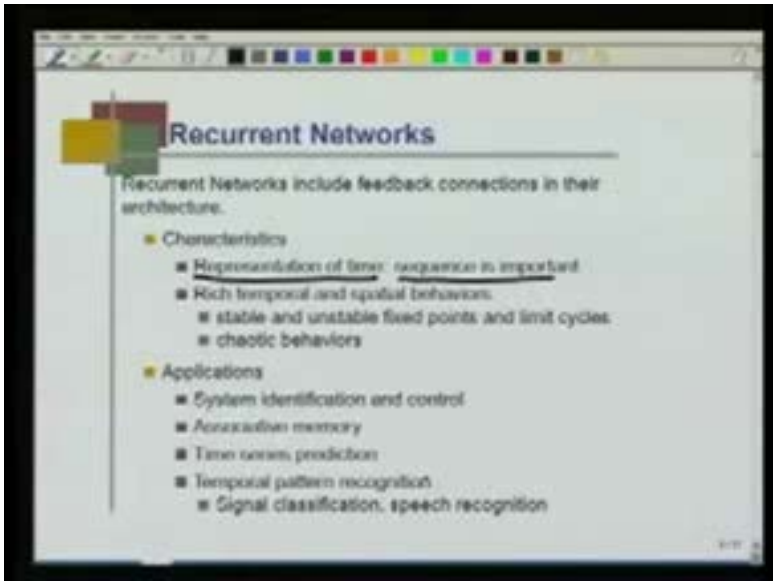
Another example of sequence is the time series prediction, any kind of time series prediction. Let us say rainfall in a year or in a decade or in a century. Let us say rainfall month-wise in a year – January, February and so on up to December. The pattern of a rainfall in a year is a sequence. This is say in mm. So, some mm in January, some mm in February, some mm in December and maybe in September, it is a maximum in a particular place. When we are dealing with patterns in which time is intrinsic, this is called temporal patterns, that is, in a year, I want to see how the pattern of rainfall is. There is a time involved in this. This particular notion of sequence is important, representation of time: sequence is important, I can explain this particular concept in another way (Refer Slide Time: 06:32).

(Refer Slide Time: 06:45)



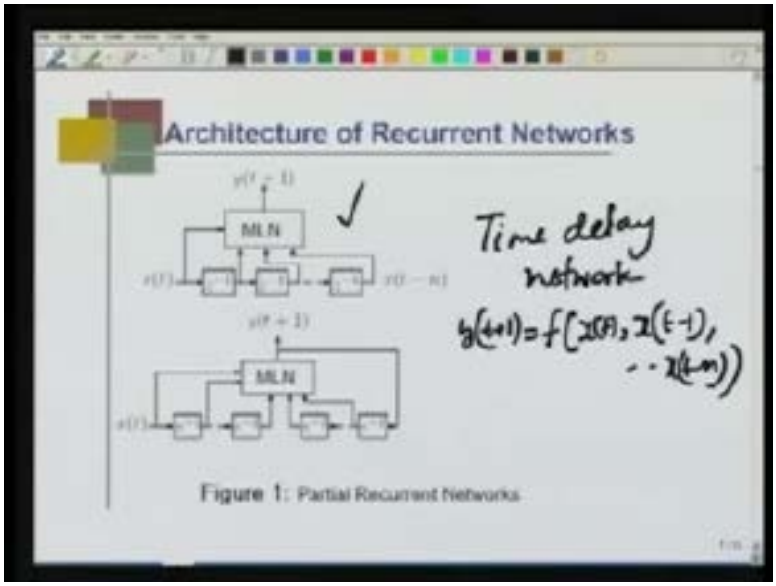
For example, I have a data pattern, this is a data set. In this data set 1, 2, 3, 4, 5, 6, the data, which are stored in this particular stack, are in a sequence. Data are stored in a sequence. If data is stored in a sequence, then when I have to represent this data to a network, I can only represent them in a sequence. In a multilayer network, if I have a data set, I can always select any of these data randomly and present to the network because the sequence is not important. I will tell this point in detail in a system identification example because this is an intelligent control course. I will explain using a relevant example in detail.

(Refer Slide Time: 08:01)



Characteristic is representation of time where sequence is important. The other thing is rich temporal and spatial behaviors, that is, this network is dynamically very rich. It has stable, unstable fixed points, it exhibits limit cycles, chaotic behaviors, and it has many other rich dynamic behavior that any dynamic nonlinear system exhibits, which a static network like a multilayer network or radial basis function network do not manifest (these networks do not manifest rich dynamical characteristics). The applications of recurrent networks are of course system identification and control, associative memory, time series prediction, temporal pattern recognition: signal classification and speech recognition. Signal classification and speech recognition are all temporal pattern recognition terms.

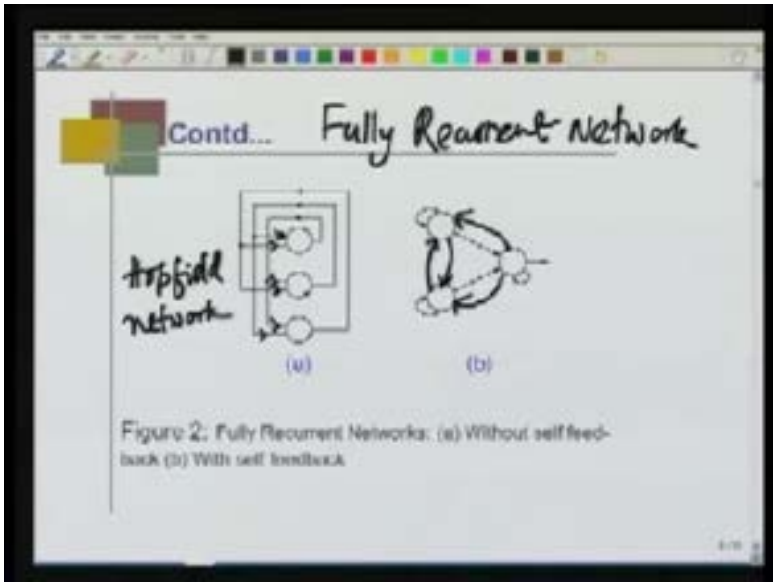
(Refer Slide Time: 09:24)



Now, we will talk about the architecture of a recurrent network. These two architectures are partial recurrent network. Partial means the main body of the network is like a multilayer network, where there are feed-forward structures layer-wise, but at the input, you can see that there is a temporal sequence. The input to the network is x t, x t minus 1, this one is x t minus 1, this is one delay further, x t minus 2 until x t minus n. This is normally known as time delay network.

Imagine we are trying to model a function like y t plus 1 is function of x t, x t minus 1 and so on. When this is the dynamic model, then this kind of network is selected (Refer Slide Time: 10:44). In many system identification examples, another kind of network is used where you see that this is a multilayer network structure – main body, but then, the input to the multilayer network in one sequence is the delayed version of x t. So there is x t, x t minus 1 up to x t minus n. Similarly, this is a delay, so the response is y t up to y t minus m. Why do we say partial recurrent? It is because the main body of the network resembles a multilayer network, the feed-forward network. This kind of network has been used particularly in system identification and time series prediction.

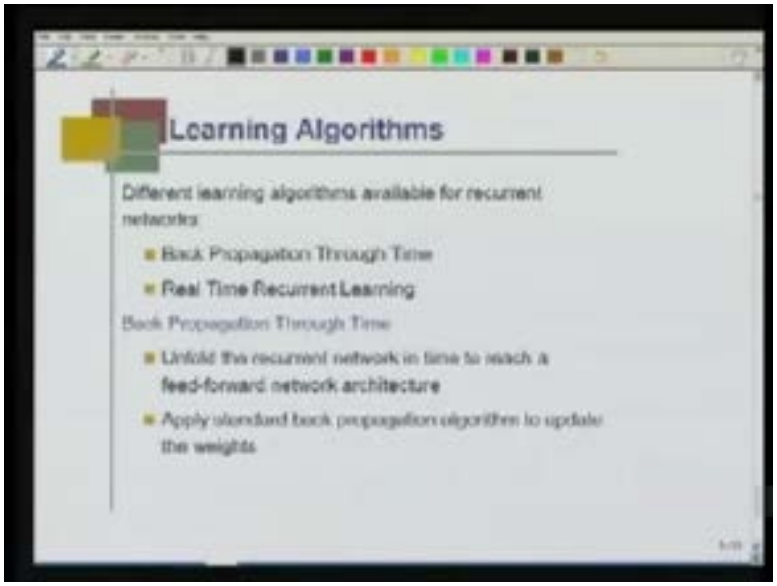
(Refer Slide Time: 11:53)



Here, these are fully recurrent networks. Earlier, we saw a partially recurrent network but this is a fully recurrent network. In a fully recurrent network, you see that the input to this neuron (Refer Slide Time: 12:13) is coming from this neuron as well this neuron. We can easily see that this one is given here and again, this one is given here. The input to this neuron is coming from the other two neurons. Similarly, this neuron's inputs are coming from this neuron and this neuron. This is this neuron and this is the other neuron and similarly, this neuron, which is this one (Refer Slide Time: 12:51), and this neuron, which is this one….

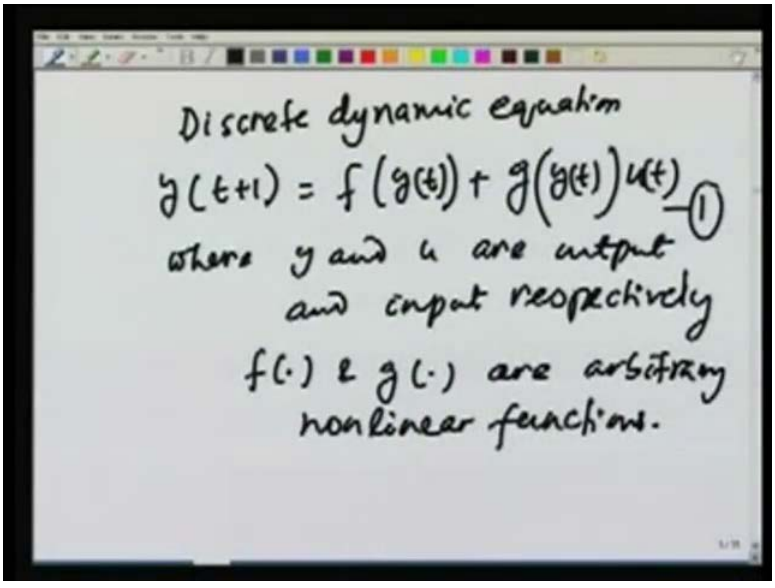
In that sense, this is a fully recurrent network. Each neuron is connected to all other neurons in the network. Normally, Hopfield network has this structure. There is a little mistake here. The branches are missing (Refer Slide Time: 13:27). This is another fully recurrent network. This network and this network are almost the same. The only difference between this network and this network is that there is a self-feedback. The output from this network is again fed back to the same unit. These two architectures are called fully recurrent. Each neuron is connected to all other neurons. One is without self-feedback and the other is with feedback. But we can have various…. These are very simplistic models that I am introducing to you. There are also many other complex models – we will not be discussing them.

(Refer Slide Time: 14:40)



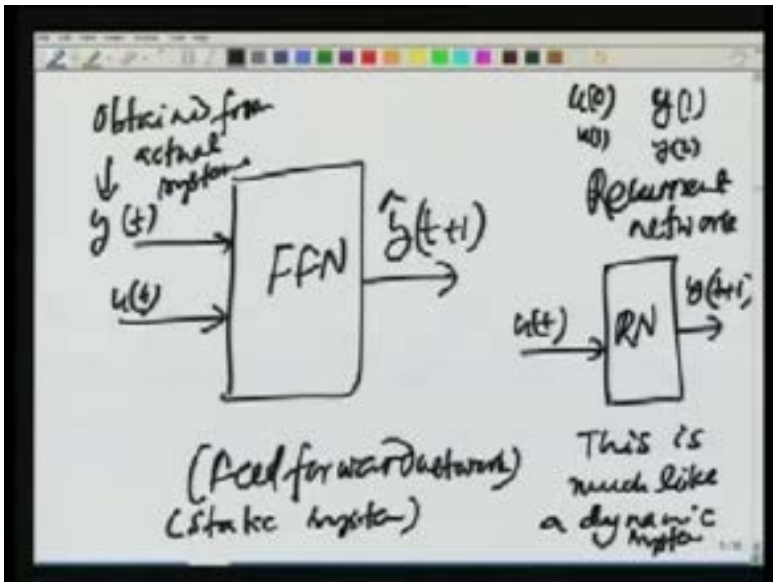
Once you have an architecture, obviously, you would like to use it. As you know, once we talk about neural network, it means the weights have to be learned, they have to be trained. There are various learning algorithms but what we will mainly focus in this class and in this course are back propagation through time, another is real-time recurrent learning. There are others also: extended Kalman filtering and many other learning algorithms that are there in the literature. We will not be taking up those things now.

(Refer Slide Time: 15:08)



Let us take a discrete dynamic equation, which is y t plus 1 is f of y t plus g of y t into u t, where y and u are output and input, respectively, and f and g are arbitrary nonlinear functions. Let us say that this particular dynamic that we have written here y t plus 1 is f y t plus g y t u t represents some nonlinear system. Now, I want to model using a feed-forward network as well as the recurrent network the dynamics of this equation, let us say this equation. If I want to use a feed-forward network, what I will do is….

(Refer Slide Time: 16:59)



In feed-forward network…. This is called a feed-forward network. My input will be y t as well as u t and output is y t plus 1. This is actually predicted output of the network (Refer Slide Time: 17:28) and y t is actually the input to the network, but is obtained from the actual system. If I really want to capture this particular dynamics that we are looking at (Refer Slide Time: 17:50) y t plus 1 is f y t plus g y t into u t using a feed-forward network, what I do is that my inputs are y t as well as u t and the output is y t plus 1 (this is the predicted one), whereas this y t is taken from the actual system.

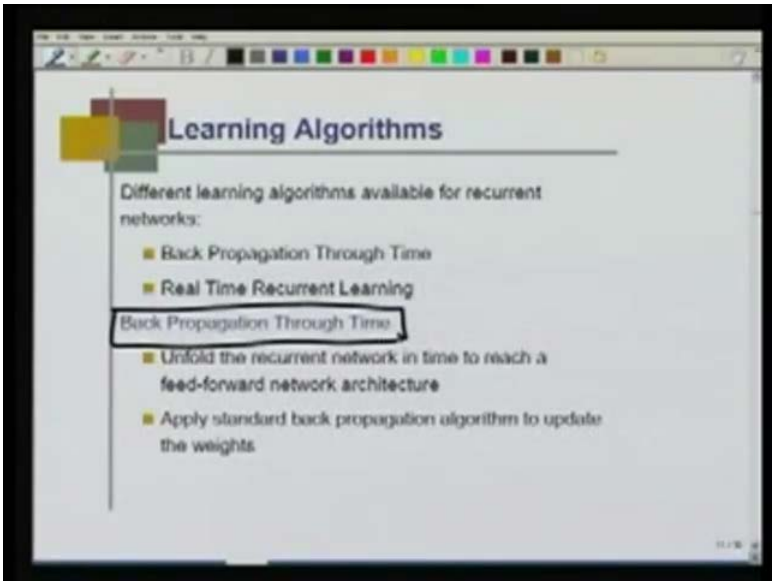
I observe the state of the system at t, y t and give an input u at t. Then, my output y t plus 1 is predicted by the feed-forward network, but a recurrent network, on the contrary, only takes the input u t and the output is y t plus 1, that is, it is possible because a recurrent network allows feedback connection from output to input, the internal states. So,

recurrent network is much like a dynamic system, whereas a feed-forward network is a static system. This is very important to understand – a feed-forward network is a static system, because….

While training a feed-forward network, we really do not worry; we really do not care about the order of the pattern. Given a pattern, that is, input/output data set, I can select any pattern, give the input of that pattern to the network, forward pass that input, find what is the output, compare with the desired one, back propagate the error, update the weight – that is the feed forward network, whereas, in a recurrent network, the sequence of data to be presented in the network must be preserved.

How will I present data to this recurrent network? u 0 y 1, then u 1 y 2, so the sequence has to be like this: u 0 y 1 first, then u 1 y 2 second and so on, but I cannot arbitrarily select any of these sequences and present to the recurrent network. There is a representation of time in a recurrent network, the sequence is important, whereas time does not play any relationship in feed-forward network – it is a static map.

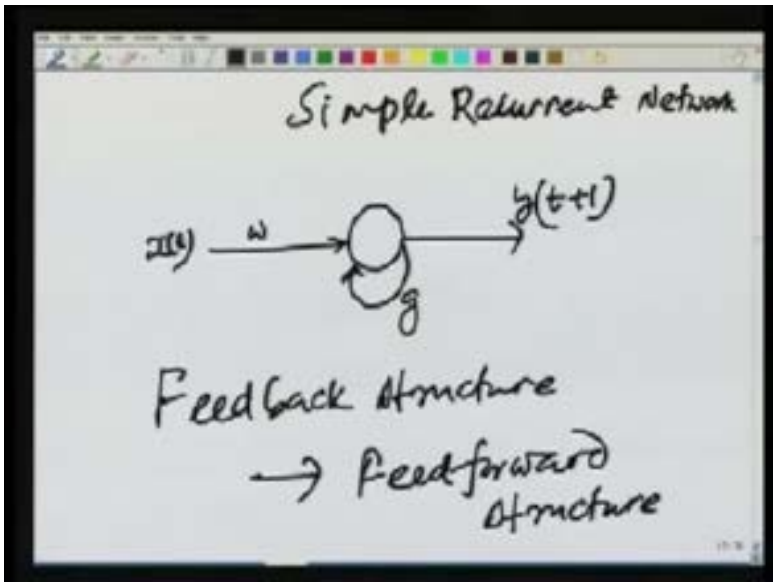
(Refer Slide Time: 21:16)



The normal training algorithms that are used for a recurrent network is back propagation through time and real-time recurrent learning. These two are very popular algorithms

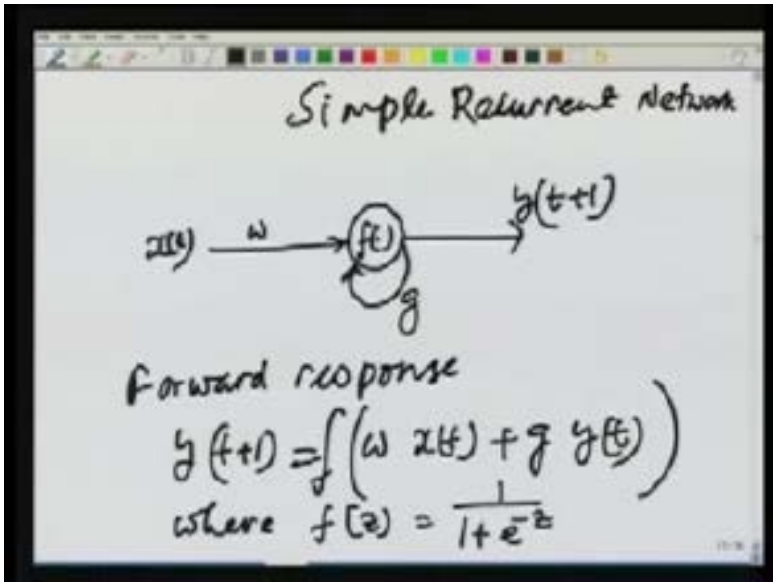
available in the literature, but there are also other like extended Kalman filtering, but in this course, we will only learn or I will only take up these two learning algorithms – back propagation through time, which is offline, and real-time recurrent learning, which is an online learning algorithm. Today, this lecture will focus on back propagation through time. In back propagation through time, we unfold the recurrent network in time to reach a feed-forward network architecture.

(Refer Slide Time: 22:05)



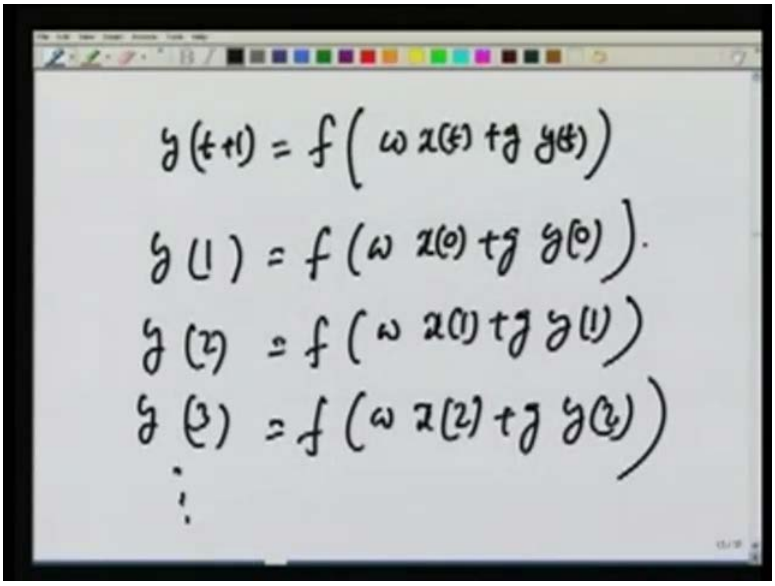
What is that unfolding? The unfolding is…. Let us take a simple recurrent network that consists of only a single neuron, the input is x t, the input to the neuron – the weight is w; there is a feedback from the output to the self-feedback and the weight is g. This is a very simple network, because it consists of only one neuron. The first part is that before we talk about unfolding this network in time…. This is the feedback structure and the objective is how to convert this feedback structure to a feed-forward structure. We want to do this. I want to convert this feedback structure to a feed-forward structure. How do I do it?

(Refer Slide Time: 23:24)



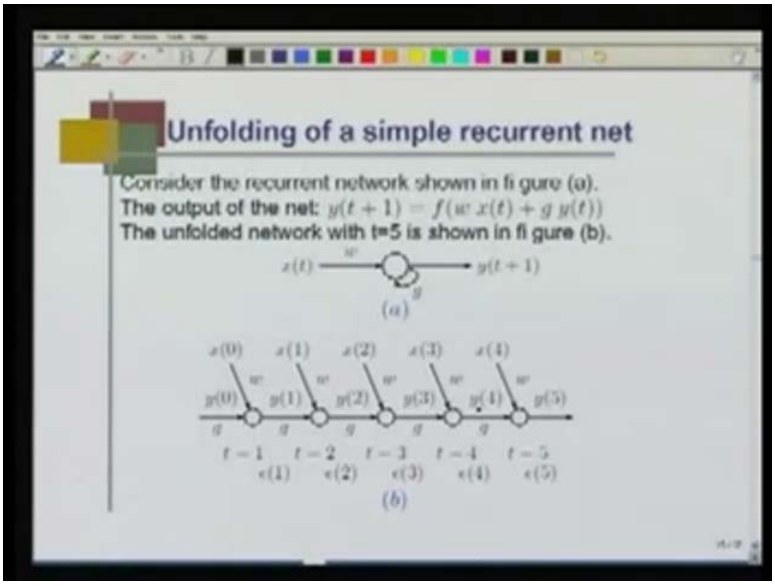
It is very easy to do. Before we can do that, we must write down the forward response of the network. What is the meaning of forward response? Given x t, what is y t plus 1? Let us say this neuron has sigmoidal activation function, that is, y t plus 1 is w x t plus g y t. You can easily see that this is the feedback, so we assume that when I do not put anything on this connection, it means that there is a delay of 1 unit. So, the y th t plus 1 sampling instant is a function of w x t, the summation of all the inputs – the w into x t plus g into y t. This is the total summation and the summed input at the input of this neuron and then sigmoidally activated, where f z is as usual… the sigmoidal activation is e to the power minus z. This is our simplest network and to this simplest network, how do you unfold this network?

(Refer Slide Time: 25:18)



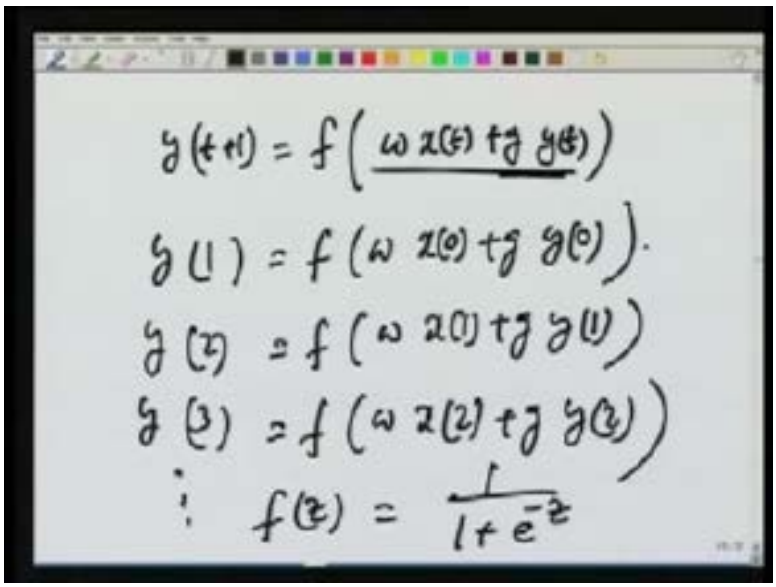
Let us see this forward response that we wrote, which is y t plus 1 is f of w x t plus g y t. This is the forward response of the simplest network that we took. If I write recursively, what do I do? I write y 1 is f of w x 0 plus g y 0. Similarly, y 2 is f of w x 1 plus g y 1. Look at this equation. I simply put t = 0, then I get the first equation; if t = 1, I get the second equation; if t = 2, then I will get the third equation and so on. We can go on like this. How do we construct a feed-forward network looking at these equations? What I will do is in my network, I create a node. In every node, I compute what is y 1 as a function of x 0 y 0, y 2 as a function of x 1 y 1 and y 3 as a function of x 2 and y 2. This is what is here.

(Refer Slide Time: 26:58)



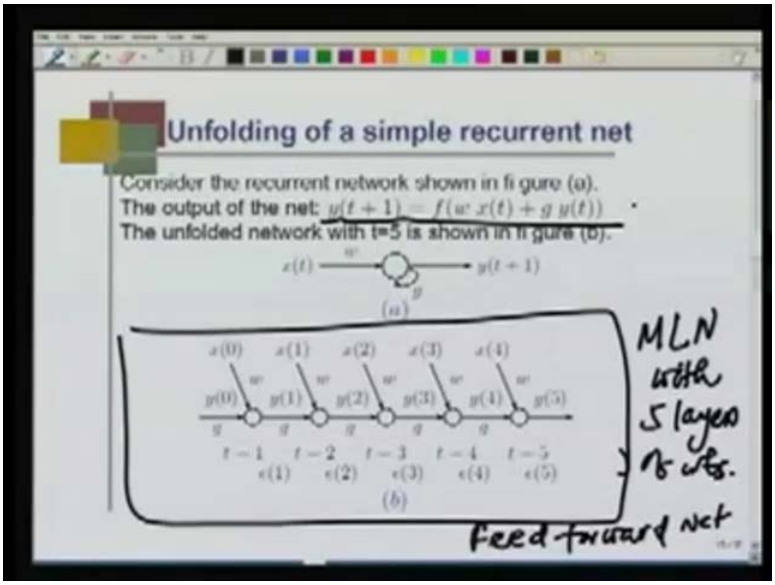
This is what we have given. Here, you see that y 1 (Refer Slide Time: 27:02) is w x 0 plus g y 0. Similarly, y 2 is w x 1 plus g y 1 and sigmoidally activated. This neuron is sigmoidally activated.

(Refer Slide Time: 27:20)



Thus, if you look at our recursive equations here, f of this summation…. (Refer Slide Time: 27:24). That means you should not forget what is the meaning of f. f is always a sigmoidal activated function. That is what we are doing here.

(Refer Slide Time: 27:40)



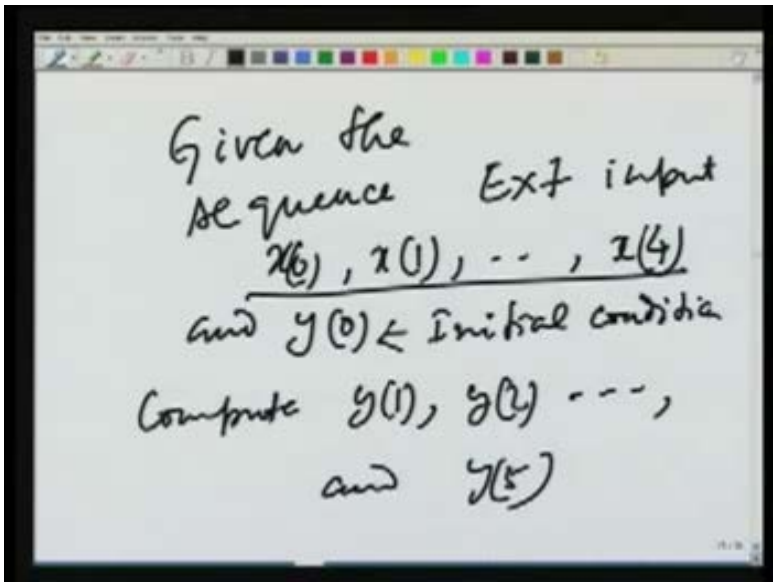
y 1 is f of w x 0 plus g y 0, y 2 is f of w x 1 g y 1 and so on. This is called rolling the network in time. Now, if you look at this network that I discussed now, if rolled over t = 5, then for t = 1, this is the first expansion (Refer Slide Time: 28:08), for t = 2, this is the second expansion. So, the first expansion y 1 in terms of y 0 x 0, the second expansion y 2 in terms of x 1 y 1, third y 3 in terms of x 2 y 2, similarly fourth y 4 in terms of x 3, y 3 and five y 5 in terms of x 4 y 4. This is how the network is unfolding over time and we can go ahead depending on the situation or demand.

Once we unfold, now this is a feed-forward network. If you look at this, this is a feed-forward network. This feed-forward network is actually MLN – multilayer network. With how many layers? I can say this is one layer, this is the first layer (Refer Slide Time: 29:12) – the first layer of weight, this is the second layer of weight, this is the third layer of weight, fourth layer and fifth layer. So, this is an MLN with five layers of weights. Given this structure, once I convert the recurrent network into a feed-forward network, the objective would be now to apply… because this is now a feed-forward network, so I

apply the back propagation through time – standard back propagation, generalized delta rule to update the weights.

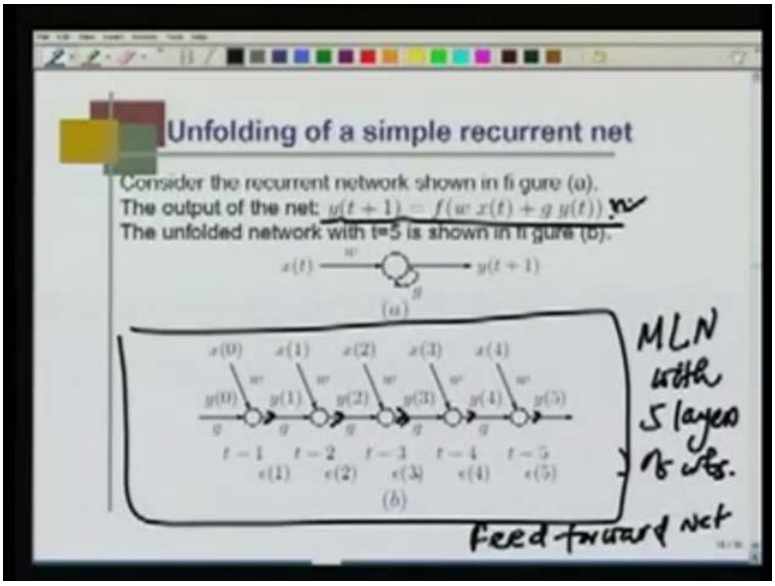
How do I do it? Now I already have the dynamic equation here (Refer Slide Time: 30:07): y t plus 1 is f w x t plus g y t. This is the dynamic equation. What I do is that given y 0 and the sequence of x 0, x 1…. These are all external input – x 0, x 1, x 2, x 3, x 4 (these are all external inputs). Given this external input and initial condition, which is y 0, let us compute what is y 1, y 2, y 3, y 4 and y 5. This is the first step.

(Refer Slide Time: 30:54)



The first step is given the sequence x 0, x 1 and so on up to x 4 and initial condition… this is the initial condition and these are external inputs, compute y 1, y 2 … y 5. This is how we generate the data.

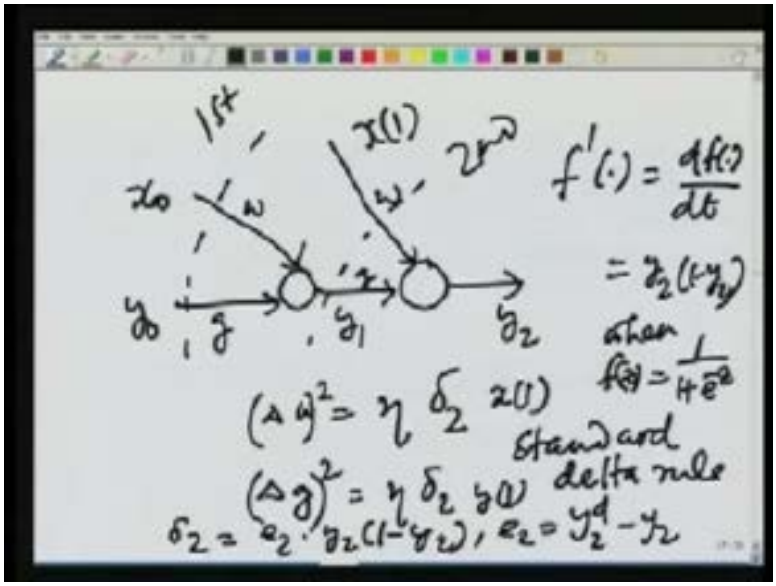
(Refer Slide Time: 31:45)



Once we generate the data, what I can do is that I now forward propagate my signal. Initial condition is y 0. Initially, this w and g are the same values but random values. Whatever w, initially, these ws are all same but all random values, some random values. Similarly, gs are all random. Then, what I do is I take the initial condition y 0 and x 0 is the input and compute what is y1; take x 1, compute y 2; take x 2, compute y 3; take x 3, compute y 4; take x 4, compute y 5. Once given x 0, x 1, x 2, x 3, x 4 and given the initial values of w and g…. These w and g values are all the same in the initial network; actually, they are the same always, because for all layers, w and g are same at any given instant; they are not different.

Given y 0 and this sequence we can compute y 1, y 2, y 3, y 4, y 5. From this model, we get the actual y 5, y 4, y 3, y 2 and y 1. Obviously, we can compute at every target – in this target e 1 (Refer Slide Time: 33:21), in this target e 2, this target e 3, this target e 4 and this target e 5. What is e 5? The response of the network is subtracted from the actual response of the network, which is y 5. This e 5 is y d 5 minus y 5, e 4 is y d 4 or y desired 4 minus y 4 and so on. Similarly, y desired 3 minus y 3 is e 3. We compute these errors at every target. Now, apply the back propagation through time. How do I do it? Let me explain in this manner.

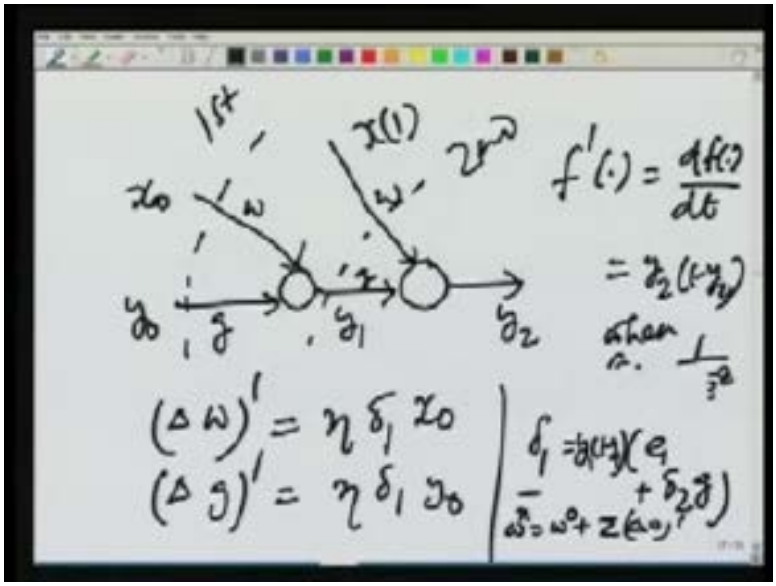
(Refer slide tine: 34:19)



Let us take a very simple thing, that is, only two layers. This is my x 0, y 0, this is y_1, this is y_2 and x_1. Obviously, this is w, this is w, this is g and this is g. What is y_2? y_2 is f of w x_1 plus g y_1. How do I update this w and this g? This is my second layer. I will write delta w 2, the change in delta w 2, using the gradient descent back propagation algorithm. How do I write this? This delta w 2 is eta delta… this is the output layer, so I put $delta_2$ – the error back propagated from the second layer, output layer and input is x 1. This is the standard delta rule. Similarly, delta g in the second layer….

I say this is the second layer (Refer Slide Time: 36:04) and this is the first layer. In the second layer, delta g due to the error from here is eta $delta_2$ and input is y 1. What is $delta_2$? $delta_2$ is obviously the error here, which is e_2 into y_2 into 1 minus y_2. What is y_2 into 1 minus y_2? We assume that f dash, which is df by dt is y_2 into 1 minus y_2, when f z is sigmoidal activation function. We have earlier mentioned this point that if f z is 1 upon 1 plus e to the power of minus z, then f dash is z into 1 minus z. That is what we are writing. What is e_2? e_2 is y_2 desired minus y_2. This is the first layer. Similarly, second layer will be again delta w in the first layer.

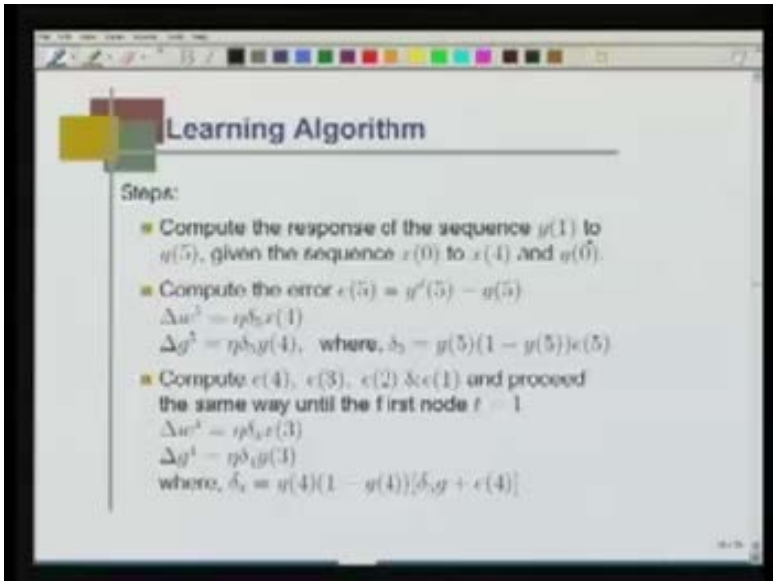
(Refer Slide Time: 37:42)



We can again do this for the first layer now. delta w 1 is eta $delta_1$ x_0 and delta g 1 is eta $delta_1$ y_0, because this is a standard delta rule. What is $delta_1$? $delta_1$ is obviously not only error back propagated from here, there is also error here (Refer Slide Time: 38:22), and that error is e_1, the error here and the back propagated error is $delta_2$, back propagated through g – that is the delta error back propagated and this is your y_1 into 1 minus y_1. So, this represents $delta_1$ (Refer Slide Time: 38:58).

Like that, using simple delta rule, given the data sequence, we can compute what is delta w 1, delta g 1, delta w 2, delta g 2 and finally add the changes in weight in every layer and sum that, and finally, w is updated as w new is w old plus sigma delta w r in every layer, because now, the same weight has been updated in every layer, but the weight is same weight. So, all those updates according to back propagation, the infinitesimally small change in weight that we get, we add over all the layers and then add to the old weight and that becomes the new weight. That is the objective here. For the network with t = 5, I just showed you t = 2.…

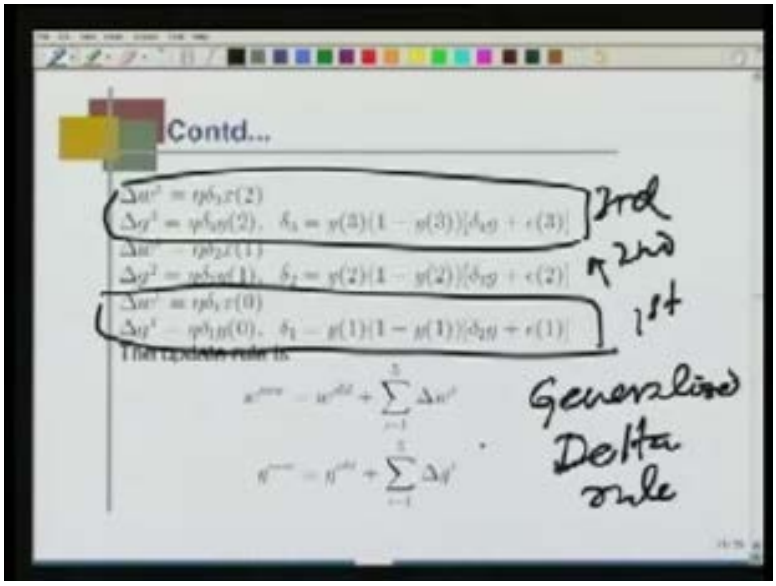
(Refer Slide Time: 40:30)



t = 5. Compute the response of the sequence y 1 to y 5, given the sequence x 0 to x 4 and y 0. We are talking about the network that we have already talked about here (Refer Slide Time: 40:45). This is my total feed-forward network constructed from the recurrent network by rolling over time and for this, we are showing the error. Computed error e 5 is y d 5 minus y 5, delta y 5 is eta delta 5 x 4, delta g 5 is eta delta 5 y 4. We can easily see here what we are trying to do (Refer Slide Time: 41:22). e 5 is y d 5 minus y 5. This delta w, the change in this weight, is obviously eta delta 5 x 4, the change in this weight g is eta delta 5 into input is y 4. That is what we are writing here (Refer Slide Time: 41:45), where delta 5 is the differentiation of the activation function or derivative of the activation function and the error at the output.

But when I come to the layer 4, my error at the layer 4 is e 4 is already there and the back-propagated error is delta 5 into g, which we will talk about now here. The error here (Refer Slide Time: 42:20) is e 4 plus back-propagated error from this layer, which is delta 5 into this weight g. Like that, we have to compute. This is for fourth layer.

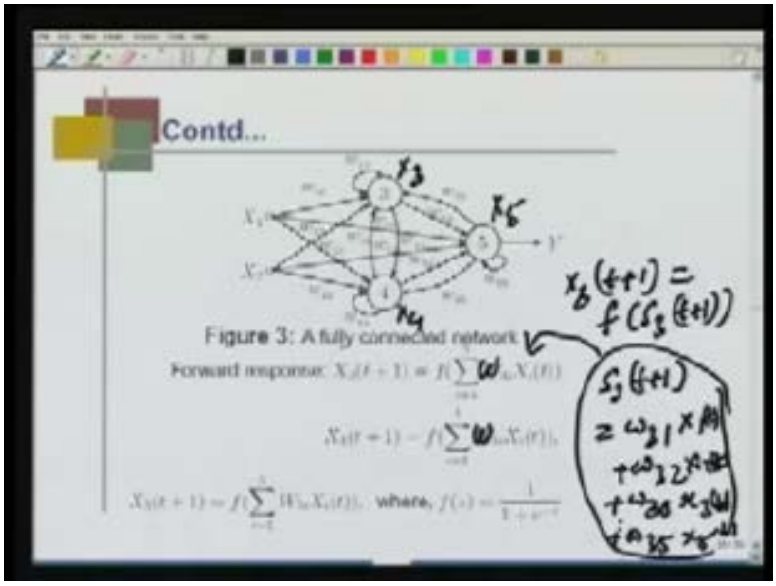
(Refer Slide Time: 42:39)



This for third layer (Refer Slide Time: 42:43), and finally, this is the first layer and this is the second layer. This is the way we apply the generalized delta rule to this feed-forward network that has been constructed from the recurrent network by rolling the recurrent network over time. Finally, after finding delta w over all the layers, delta g over all layers, what do we compute? w new equal to w old and we add over all layers delta w.

Similarly, g new is g old plus delta g i over all layers; we add them because there are five layers of weights, we add and then we find g new. We talked about a simple network. Now, let us go to a little complicated network. Is a complicated network very difficult to roll over time? No, it is not. Once we understand the principle, rolling over time is not very difficult.

(Refer Slide Time: 44:14)



For example, you see here, this is a fully connected network, we have three neurons and two external inputs X_1 and X_2 and one output is Y. Let us say the output of this neuron is this is X_3, this is X_4, this is X_5. So, Y is X_5. If I look at the forward response, if you look at the normal convention, if you look at the weight connections, the weights that are being connected here, this weight from the input 1…. You see the connection X_1, X_2, X_3, X_4 and X_5.

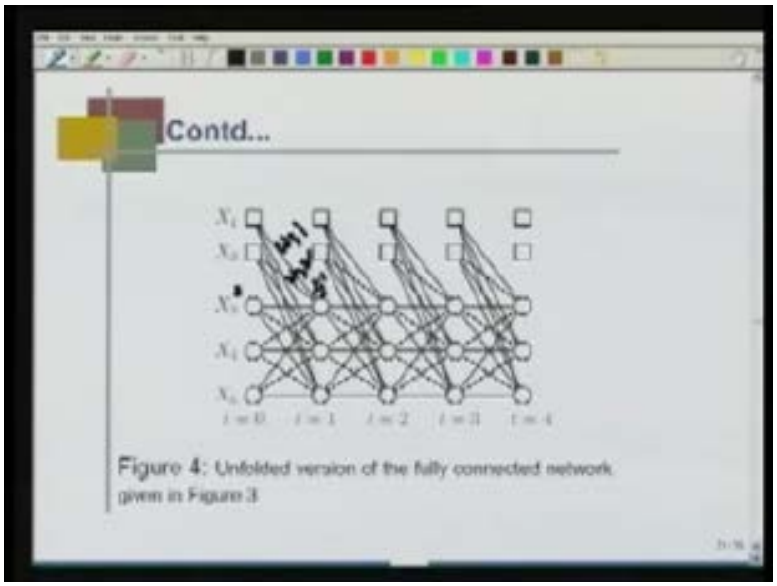
Convention-wise, the weights are…. This is the node to which the connection is coming from 1. Obviously, the weight that has been named is w_{31}. w_{31} means the weight that connects the node 3 from node 1. Similarly w_{33}, by just looking at this weight, you can say this is a self-feedback weight, that is, this weight connects node 3 to itself. Similarly, w_{35}, you can see that this weight is coming from the node 5 and connecting to 3. Similarly, w_{45}, if you look at this one, this is coming from 5 and connecting 4. Similarly, w_{55} is a self-feedback weight.

This is the convention of designating the weights and this is a fully connected network. What is the forward response? Forward response is what the output of each neuron is given X_1 and X_2 and the present state X_3 t, Y_5 t and X_4 t. So, X_3 t plus 1 is the response of the node 3 at the next time instant is a sigmoidal activation function f of the

summation of w_{3i} X_i t. Similarly, X_4 t plus 1 is w_{4i} X_i t and X_5 t plus 1 is w_{5i} X_i t, because each node is connected to every node – all the five nodes (two input nodes and three neurons, total 5 nodes).

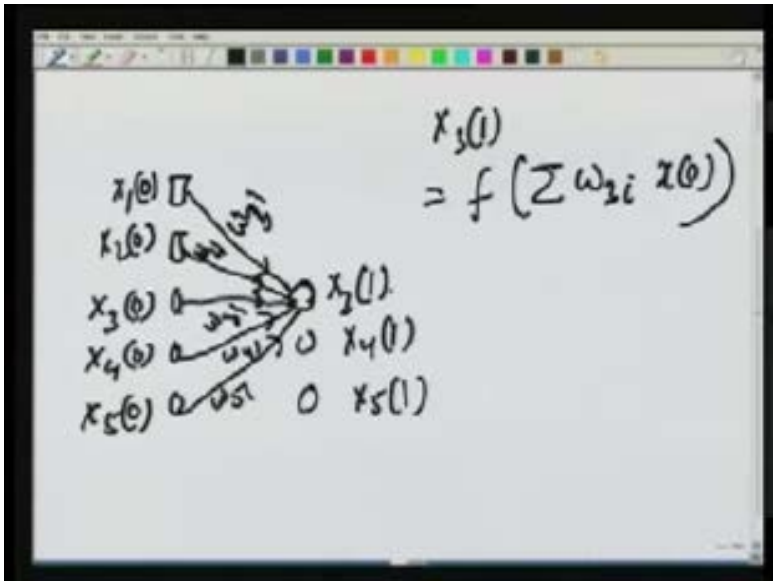
You can easily see that X_3 t plus 1 is function of S_3 t plus 1. What is S_3 t plus 1? S_3 t plus 1 is summation that is w_{31} X_1 plus w_{32} X_2 plus w_{33} X_3 and so on w_{35} X_5. We are inserting this particular item here (Refer Slide Time: 48:08), which is known as w_{3i} X_i t. This has to be t (Refer Slide Time: 48:15). Time has to be there; we cannot do away with time in a recurrent network like we have done in the feed-forward network.

(Refer Slide Time: 48:29)



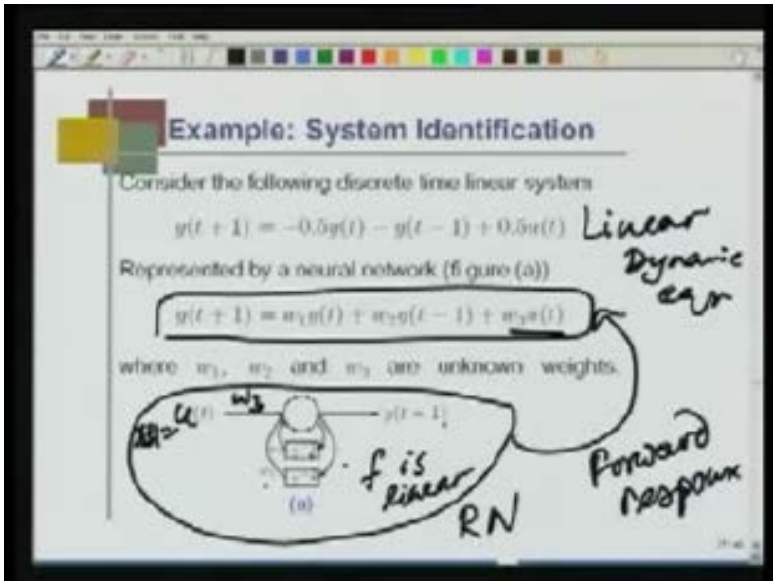
For the same network, once I know this forward response, I can easily write down here, I can unfold the same network in time here. Obviously, this is my X_3 0, this is X_3 1. So, X_3 1, X_3 at time 1 is a function of X_1 and this is w_{31}; for X_2, this has to be w_{32}; this is for X_3. The best way to look at it is like this.

(Refer Slide Time: 49:09)



I have two input nodes and this is the initial state. This is X_1 0, X_2 0 and these are all initial states. Based on this initial state, the new states are X_3 1, X_4 1 and X_5 1. How do I compute X_3 1? X_1 0 w_{31} is the weight connecting X_1 0. From X_2 0, this is w_{21}. From X_3 0, it is w_{31}, from X_4 0, it is w_{41} and from X_5 0, it is w_{51}. The forward response X_3 1 is a function of summation w_{3i} x 0. Based on that, we found out the connection between X_3 1, and these initial states in the previous instant. Similarly, X_4 1 and X_5 1. That way, we can roll this network over time (Refer Slide Time: 50:53).

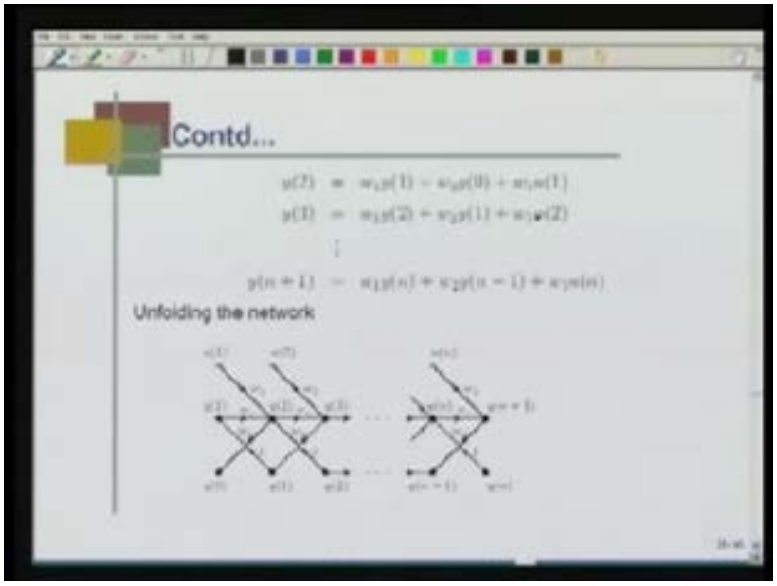
(Refer Slide Time: 50:57)



Let us take one example today to illustrate what we have learned. Let us take a simple linear network, linear dynamics. y t plus 1 is minus 0.5 y t minus y t minus 1 plus 0.5 u t –a simple discrete dynamic equation. It is a linear dynamic equation, discrete dynamic. We can represent this particular equation in terms of this (Refer Slide Time: 51:37). y t plus 1 is w_1 y t plus w_2 y t minus 1 plus w_3 u t, where w_1, w_2 and w_3 are unknown weights (we do not know) and this is the recurrent network to represent that.

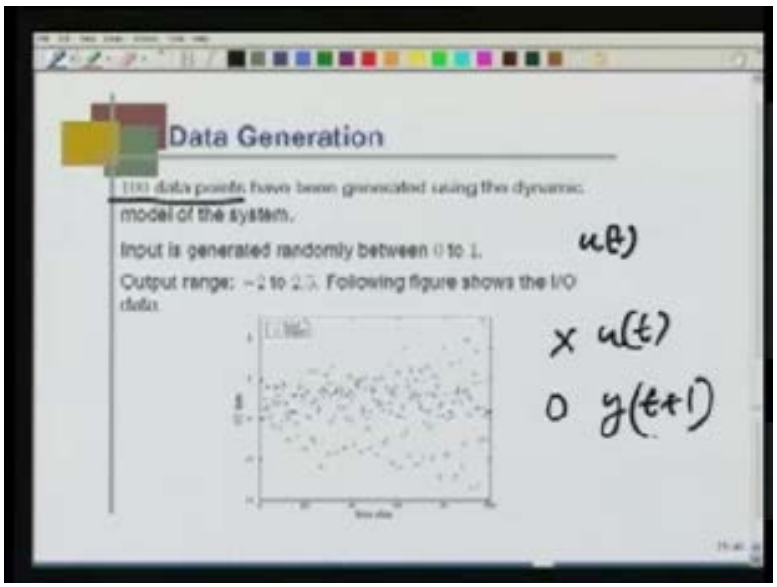
We give x t, output is y t plus 1 and you see that there are two feedback connections here, self-feedback with one delay and this is two delay. Obviously, if I write the forward response of this network, then this is the forward response (Refer Slide Time: 52:25). The forward response is y t plus 1 is x t into this is w_3 x t into w_3, this quantity (Refer Slide Time: 52:42). Actually, we say x t but in this case, it is u t (x t is u t). So, x t into w_3 or u t into w_3 plus w_1 into y t is one delay and plus w_2 into y t minus 1 is two delay. This is my recurrent network and activation function here f is linear. So f is linear.

(Refer Slide Time: 53:23)



If I take this network, similarly from the forward response, I can write what is y 2. y 2 is w_1 y 1 plus w_2 y 0 plus w_3 u 1 and y3 is w_1 y 2 plus w_2 y 1 plus w_3 u 2 and so on. This is the corresponding unfolding in time.

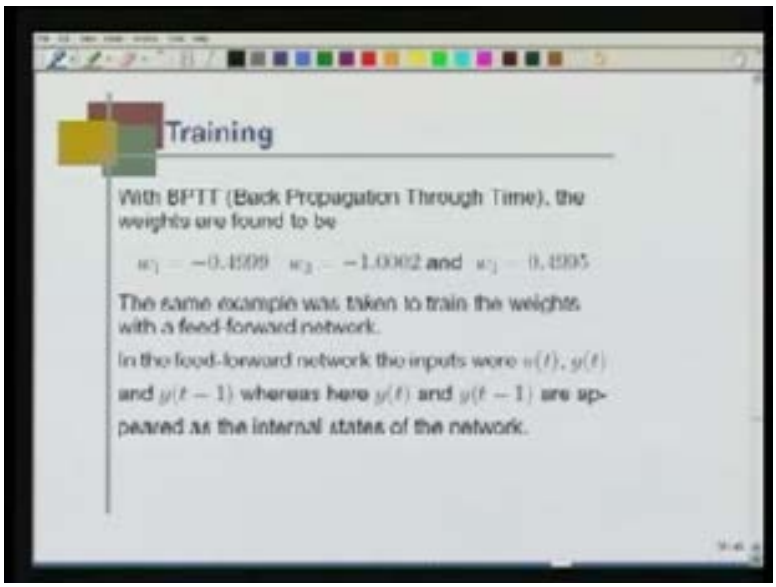
(Refer Slide Time: 53:47)



We apply this back propagation through time using generalized delta rule. We generated 100 data points, where the input u t was selected randomly between 0 and 1, and given

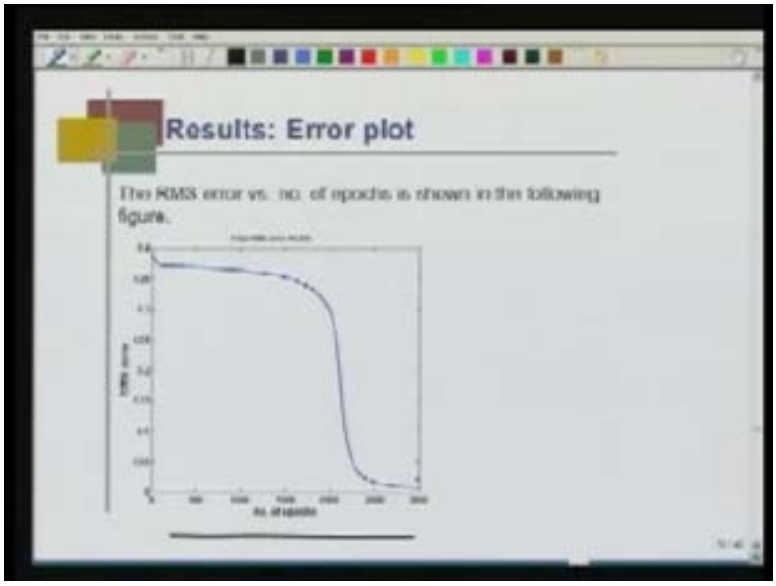
the dynamic equation that we represented, we gave for data generation, we used this particular equation for data generation (Refer Slide Time: 54:21). If you look at this data, the circles represent the output, the cross is u t and the circle is y t plus 1 (Refer Slide Time: 54:39). This is the input data generated.

(Refer Slide Time: 54:44)



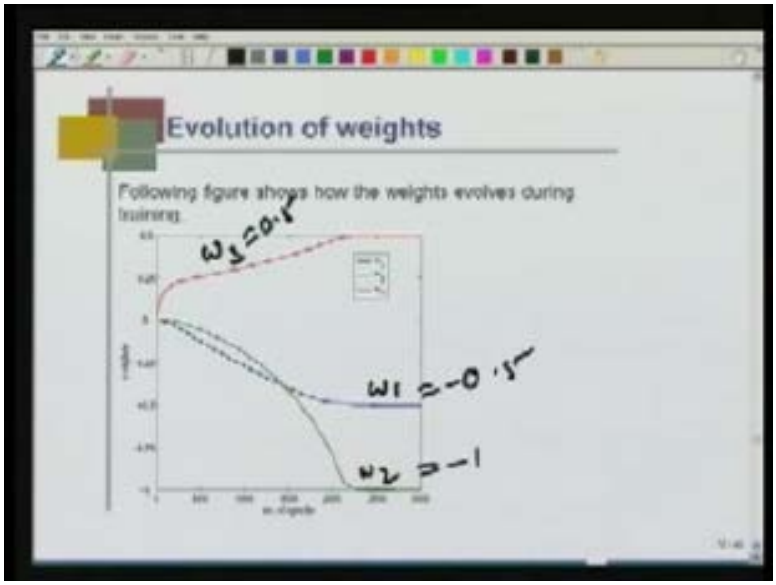
Then with back propagation through time, we trained it. When you trained, let us see how the training took place.

(Refer Slide Time: 54:56)



This is RMS error over number of epochs. You can see how the RMS error reduced. There were almost 3,000 epochs.

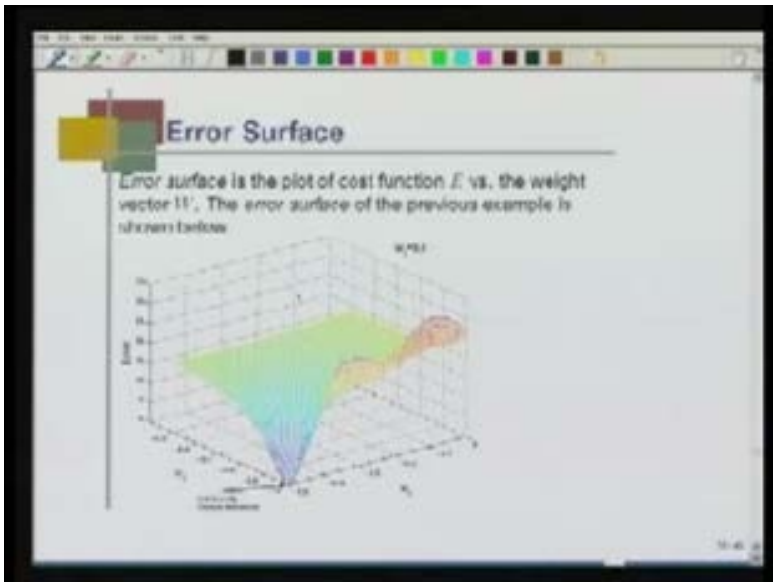
(Refer Slide Time: 55:14)



These are the weights. This is our w_3, this is our w_1, and this is our w_2. w_3 has reached 0.5, w_1 has reached minus 0.5, and w_2 is —1. If I go back and look at the equation that

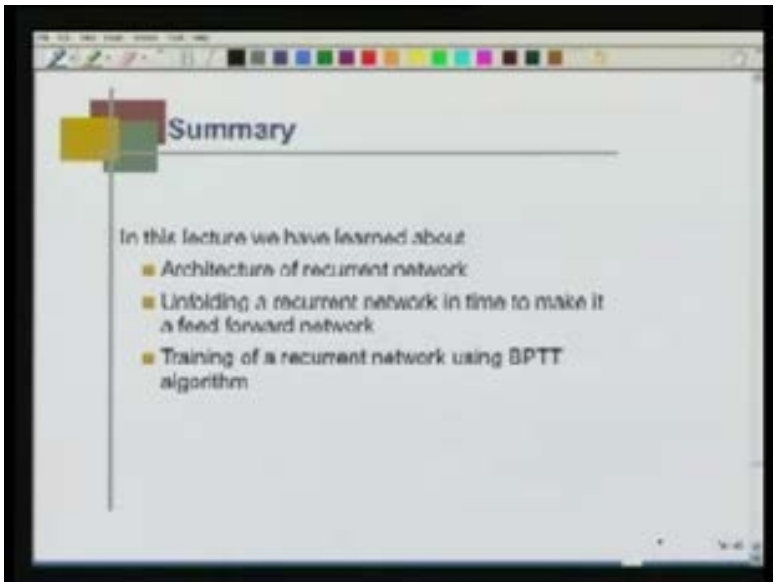
was given to us (Refer Slide Time: 55:47), you can see this was my w_3, which is 0.5, w_2 also I got —1 and w_1 is —0.5.

(Refer Slide Time: 55:56)



The error surface is the plot of cost function E versus the weight vector W. You can see how finally we reached the global minimum. Since this is the linear dynamics, we can easily reach the global minimum and that is not a problem.

(Refer Slide Time: 56:16)



To conclude, in this lecture, we learned about the architecture of a recurrent network, unfolding a recurrent network in time to make it a feed-forward network, training of a recurrent network using BPTT algorithm and we showed an example of a system identification of a linear discrete dynamic. Using BPTT, we showed that the actual parameters are learned. Thank you very much.