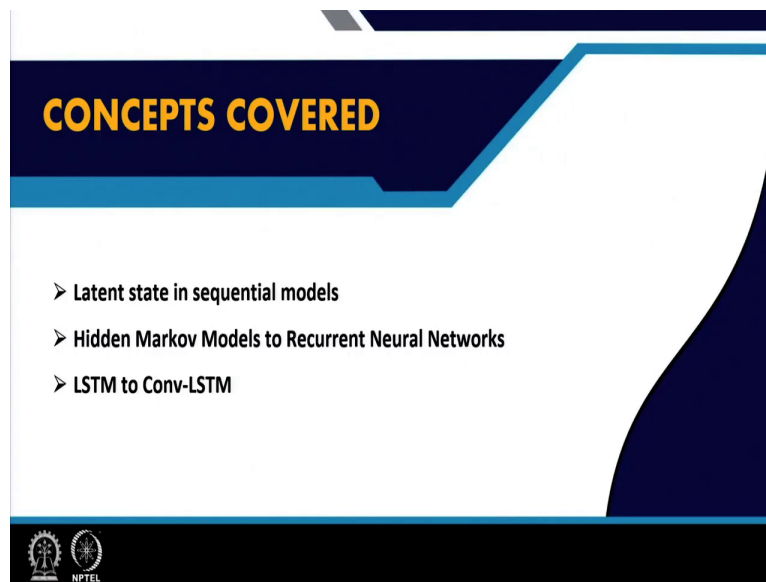


Machine Learning for Earth System Sciences
Prof. Adway Mitra
Department of Computer Science and Engineering
Centre of Excellence in Artificial Intelligence
Indian Institute of Technology, Kharagpur

Module - 02
Machine Learning Review
Lecture - 14
Sequential Models for Temporal Problems

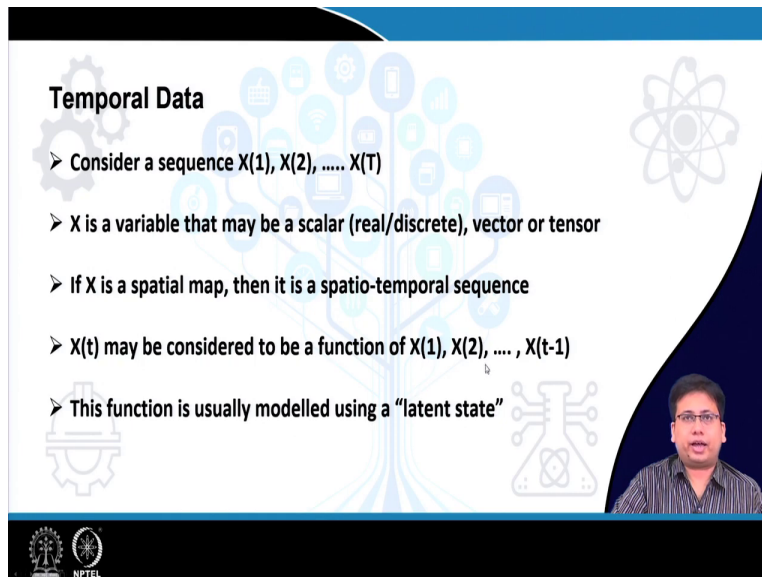
Hello everyone. Welcome to lecture 14 of this course on Machine Learning for Earth System Science. We are right now in module 2 which is about the Review of Machine Learning techniques relevant for this course. So, today we will talk about Sequential Models for Temporal Problems in earth sciences.

(Refer Slide Time: 00:43)



So, the main concepts to be covered in this lecture are latent state in sequential models, the hidden Markov models to recurrent neural networks and from LSTMs to Conv-LSTMs.

(Refer Slide Time: 00:55)



Temporal Data

- Consider a sequence $X(1), X(2), \dots, X(T)$
- X is a variable that may be a scalar (real/discrete), vector or tensor
- If X is a spatial map, then it is a spatio-temporal sequence
- $X(t)$ may be considered to be a function of $X(1), X(2), \dots, X(t-1)$
- This function is usually modelled using a "latent state"

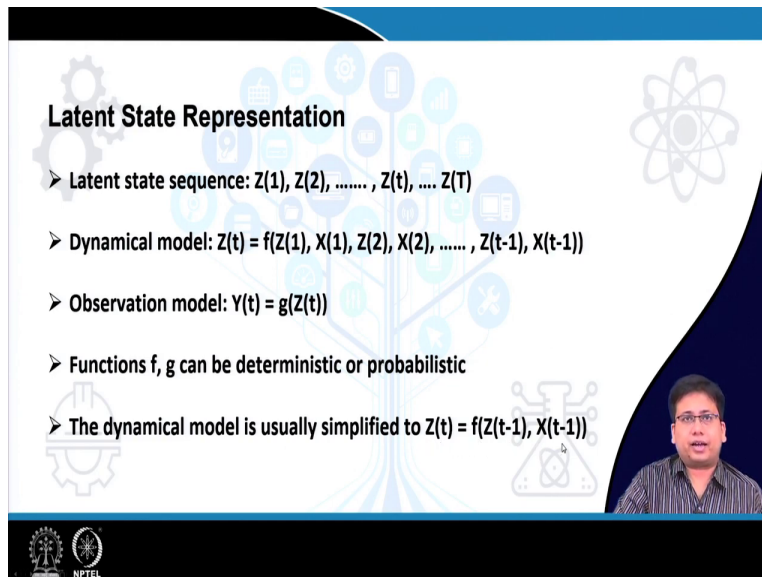
The slide features a background with various icons related to data science and technology, including a gear, a lightbulb, a smartphone, a laptop, a bar chart, a pie chart, a network diagram, and a molecular structure. A video inset in the bottom right corner shows a man with glasses speaking. The NPTEL logo is visible in the bottom left corner.

So, first of all let us talk about temporal data. So, like here we have in the temporal data we can also consider as sequential data. So, like here the data is comes in a sequence like this $X(1), X(2), \dots$. Now, the this is different from normal data set in the sense that these they must these 1 2 etcetera these are not just arbitrary serial numbers, but they define some kind of a sequence that is data $X(2)$ follows data $X(1)$, data $X(3)$ follows data $X(2)$ and so on.

Now this X this is a variable, it can be a scalar that is either real or discrete or it can be a vector or matrix tensor whatever. Now, if X is some kind of a spatial map like the let us say something like an image, the kind of or the kind of spatial data which we discussed in the last lecture then it is basically a sequence of spatial data. So, like we can call it as a spatio temporal sequence. And this $X(t)$ we can consider it to be a function of the past observations $X(1), X(2), \dots, X(t-1)$.

Now, this kind of function which connects $X(t)$ to the previous observations; this is usually modeled with the help of a latent state, that is of course, we do not have access to the we do not have anything like we only have the data we do not have a like anything like what produces the data, but we imagine that there is a system which is generating the data and the system we imagine it is having some kind of a latent state like which is defined at every time point or and that latent state is driving the observations.

(Refer Slide Time: 02:48)



Latent State Representation

- Latent state sequence: $Z(1), Z(2), \dots, Z(t), \dots, Z(T)$
- Dynamical model: $Z(t) = f(Z(1), X(1), Z(2), X(2), \dots, Z(t-1), X(t-1))$
- Observation model: $Y(t) = g(Z(t))$
- Functions f, g can be deterministic or probabilistic
- The dynamical model is usually simplified to $Z(t) = f(Z(t-1), X(t-1))$

The slide features a background with various icons related to technology and science, including gears, a smartphone, a laptop, a network diagram, and a molecular structure. A video inset in the bottom right corner shows a man with glasses speaking. The NPTEL logo is visible in the bottom left corner.

Now, so, we basically we have a sequence of latent states like this $Z(1), Z(2), \dots, Z(t), \dots, Z(T)$. So, like so, parallel to the sequence of observations $X(1), X(2)$ we have this sequence of latent state which of course, we can we like we do not know its values because it is latent these like it is like we just imagine this is to be there.

Now the dynamical model is like when we define how this state it changes over time taking by taking as input the all the previous state values as well as the previous observations. So, when we are defining $Z(t)$. So, we just imagine that there is a function f which is which takes as inputs all of them.

$$Z(t) = f(Z(1), X(1), Z(2), X(2), \dots, Z(t-1), X(t-1))$$

Now, of course, these function f can be defined in different ways in different models and of course, not all of these inputs are going to be used. Different models use only different parts of these like of all these previous data.

And then there is an observation model also which is like. So, X is like what we can like we can say this is the X are the inputs now corresponding to that we can have outputs also. So, $Y(t)$ the outputs we can say these are what we get out of the system that is like we define it in terms of

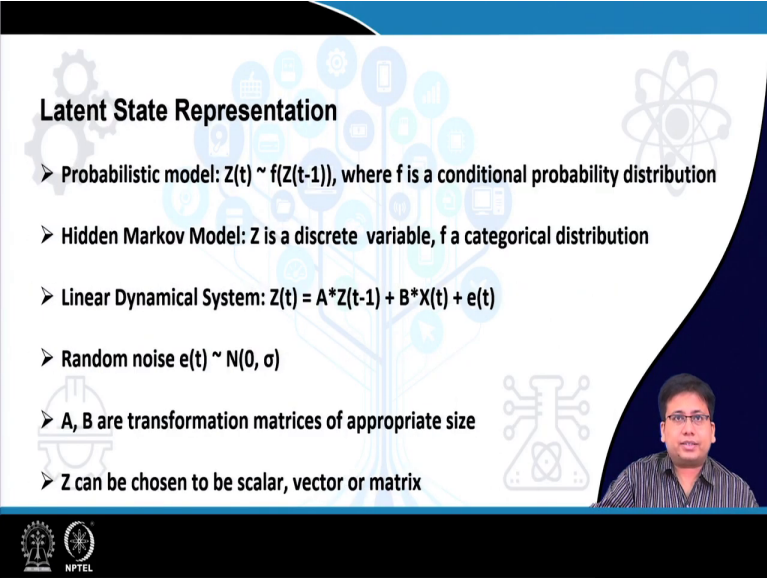
another function g . And the output at any given state $Y(t)$ like this is usually considered to be a function of the current system state only.

$$Y(t) = g(Z(t))$$

Now, these functions f or g this can be either deterministic or probabilistic and in either case like they can be defined in various ways and this dynamical model which we mentioned it is usually simplified to that $Z(t)$ the current system state is just a function of the current input $X(t - 1)$ and the previous sorry I am sorry this will be $X(t)$ not $X(t - 1)$ and $Z(t - 1)$ that is the previous state. That is the current state is a function of the previous state and the current current input.

$$Z(t) = f(Z(t - 1), X(t - 1))$$

(Refer Slide Time: 05:04)



Latent State Representation

- Probabilistic model: $Z(t) \sim f(Z(t-1))$, where f is a conditional probability distribution
- Hidden Markov Model: Z is a discrete variable, f a categorical distribution
- Linear Dynamical System: $Z(t) = A * Z(t-1) + B * X(t) + e(t)$
- Random noise $e(t) \sim N(0, \sigma)$
- A, B are transformation matrices of appropriate size
- Z can be chosen to be scalar, vector or matrix

The slide features a blue header and footer with various icons (gears, atom, lightbulb, circuit) and a small video inset of a speaker in the bottom right corner. The NPTEL logo is visible in the bottom left corner.

So, like one there are multiple models which work on this kind of framework. So, this is the in the case of the probabilistic model $Z(t)$ is just considered to be as like a conditional probability distribution on using $Z(t - 1)$ and may be $X(t)$ if it is there. So, if so, in case of hidden Markov model which is the most famous probability probabilistic model. So, there the X is not present. There is no concept of the input X there is just Z .

$$Z(t) \sim f(Z(t-1))$$

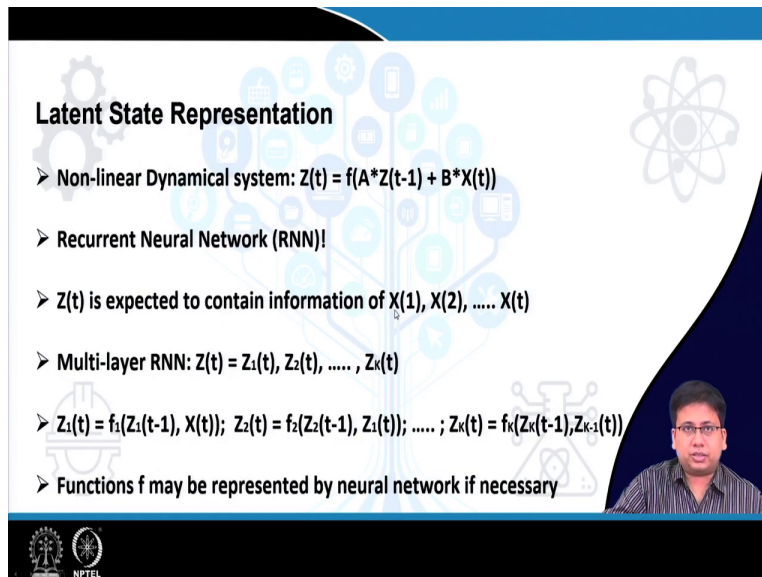
So, in that case in case of hidden Markov model $Z(t)$ is just like it is a discrete variable and this f is a categorical distribution. So, $Z(t)$ is a discrete variable which follows a categorical distribution taking $Z(t-1)$ as the input and. Now then there is another kind of model called the linear dynamical system. So, there Z is no longer bound to be discrete, but rather $Z(t) = A * Z(t-1) + B * X(t) + e(t)$.

So, where $X(t)$ is the current input and $Z(t-1)$ is the past system state. And $e(t)$ is some kind something like a random noise and A and B are transformation matrices of the appropriate size.

So, like depending on what this Z is whatever this Z might be a scalar or it may be a vector or it can be a matrix or something like that. So, these A and B like they are defined in a particular way like. So, that this the whole thing is consistent with the log of matrix multiplication and this $e(t)$ this is also some kind of a this is a random noise which is of the same size as $Z(t)$.

So, if $Z(t)$ is a scalar then $e(t)$ will also be a scalar and if $Z(t)$ is let us say is a vector or a matrix then $e(t)$ will also be that except that every element of $e(t)$ in that case we may assume to be following a random noise like this ok. Now, this yeah so, this Z as I said like exactly what form it will take depends on the exact problem which we are trying to solve.

(Refer Slide Time: 07:21)



Latent State Representation

- Non-linear Dynamical system: $Z(t) = f(A*Z(t-1) + B*X(t))$
- Recurrent Neural Network (RNN)!
- $Z(t)$ is expected to contain information of $X(1), X(2), \dots, X(t)$
- Multi-layer RNN: $Z(t) = Z_1(t), Z_2(t), \dots, Z_k(t)$
- $Z_1(t) = f_1(Z_1(t-1), X(t)); Z_2(t) = f_2(Z_2(t-1), Z_1(t)); \dots; Z_k(t) = f_k(Z_k(t-1), Z_{k-1}(t))$
- Functions f may be represented by neural network if necessary

The slide features a blue header and footer. The background is white with faint icons of a gear, a lightbulb, a smartphone, and a brain. A small video inset in the bottom right corner shows a man with glasses speaking. The NPTEL logo is in the bottom left corner.

Now instead of a linear dynamical system we can also have a non-linear dynamical system. And that is forms the basis of what is known as the recurrent neural network. So, the so, like instead of having a just a linear function, I can apply some kind of non-linearity by putting in some suitable non-linear function like this.

Now, this $Z(t)$, the system state this is expected to contain the information of the all the past inputs $X(1), X(2), \dots$. So, $Z(1)$ like we as we can understand $Z(1)$ takes in $X(1)$ and is a representation of $X(1)$ now $Z(2)$ it takes in $Z(1)$ as well as $X(2)$, but $Z(1)$ contains we can say $Z(1)$ contains $X(1)$. So, $Z(2)$ will contain $X(1)$ as well as $X(2)$.

Similarly, $Z(3)$ again will contain $X(1), X(2), X(3)$ which will receive at the input and so on. So, we can also define a multilayer RNN like this that is instead of having just a single like system state Z we can actually have like several state variables like $Z_1(t), Z_2(t), \dots, Z_k(t)$ and these though and these are also defined sequentially that is first $Z(1)$ wherever the input comes in.

So, $Z_1(t)$ is defined in terms of the past value of $Z(1)$ and the current input $X(t)$.

$$Z_1(t) = f_1(Z_1(t-1), X(t))$$

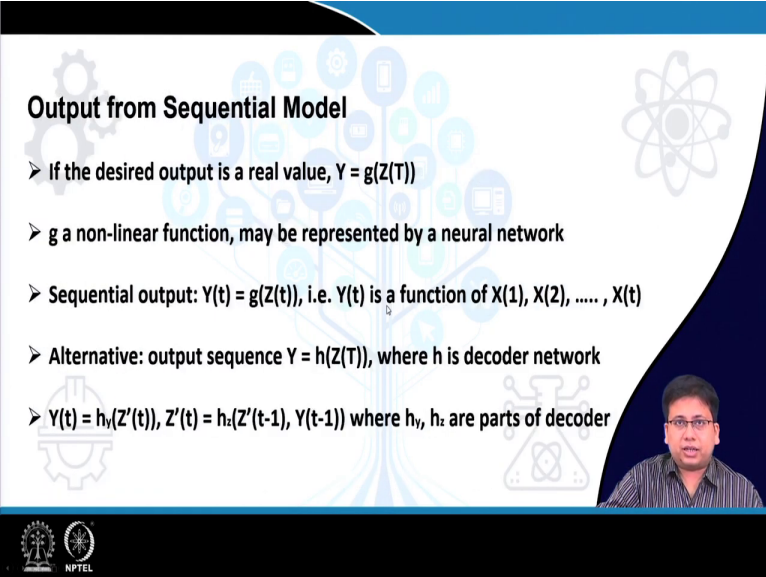
Now that $Z(1)$, once it is computed that value of $Z(1)$ is becomes input to the next layer where it is $Z(2)$.

$$\text{So, } Z_2(t) = f_2(Z_2(t-1), Z_1(t)) .$$

Now once this $Z(2)$ is calculated now that it will be passed on to the third layer where $Z(3)$ at time t will be calculated and so on.

So, let in that way we can say that we finally, have the $Z(k)$. So, the system state in this case now consist of k variables like this and this functions f these can also be arbitrarily complex functions. In fact, they can be these functions can themselves be represented as something like a neural network.

(Refer Slide Time: 09:46)



Output from Sequential Model

- If the desired output is a real value, $Y = g(Z(T))$
- g a non-linear function, may be represented by a neural network
- Sequential output: $Y(t) = g(Z(t))$, i.e. $Y(t)$ is a function of $X(1), X(2), \dots, X(t)$
- Alternative: output sequence $Y = h(Z(T))$, where h is decoder network
- $Y(t) = h_v(Z'(t))$, $Z'(t) = h_z(Z'(t-1), Y(t-1))$ where h_v, h_z are parts of decoder

The slide features a blue header and footer with various icons (gears, atom, brain, etc.) and a video inset of a man speaking in the bottom right corner. The NPTEL logo is visible in the bottom left corner.

So, we know already know that neural networks can be used to represent various functions. Now next comes the question of output. So, X is the input, Z is the system state now what is the output? So, why we can denote it as an output? So, there the output can be anything like it can be a single real value a scalar. So, it is like a that is a situation when the input is a sequence $X(1), X(2), \dots, X(T)$ and the output is just one value Y .

So, in that case we can express $Y = g(Z(T))$ which is the final system state that is the final when the final system state is achieved when the entire input sequence has been received. So, the $Z(T)$ in a sense contains the entire input information is represented in some arbitrary way and then on that I apply the function g and get the output.

Now, this g can be a non-linear function and it can again be represented as another neural network. On the other hand instead of having just a single scalar as the output we can have a sequential output like this. So, we can at every time point $Y(t)$ I can expect a sequence an output like this $Y(t)$ which is the like which is a representation of the current system state that is $Z(t)$.

So, is now $Z(t)$ the current system state is basically some representation of all the input received so far. So, in that case the output I am receiving $Y(t)$ at any given point of time is a function of the input that has been received till then and an alternative to this is like we instead of generating the sequence or the or the outputs instantaneously we first absorb the entire sequence and get $Z(T)$ which is the we can say is some kind of a intermediate representation of the entire sequence.

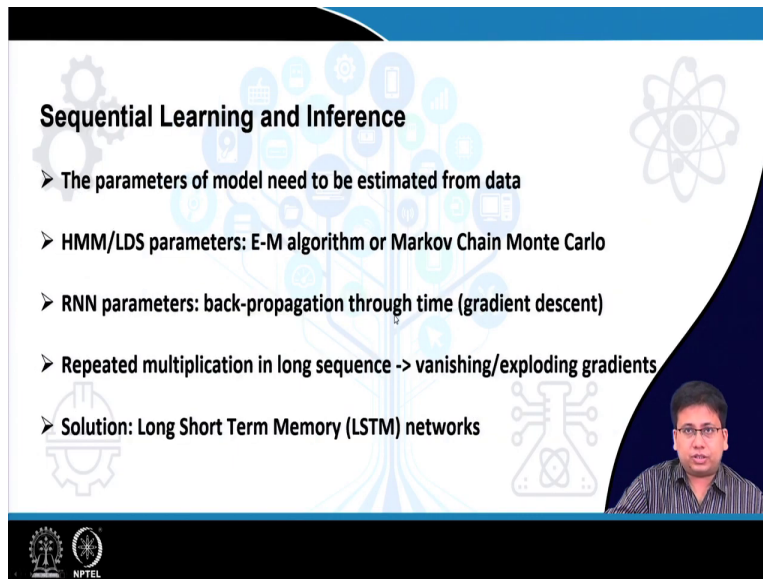
Then we define some kind of a like a decoder network which we denote as h . So, that h works on that intermediate representation $Z(T)$ to produce an output sequence Y . So, like it is like saying that we have this like this Z' is something like the output system state or the decoder system state it starts with $Z(T)$ with the that is the representation of the input and then the like in the output we have another sequential model and another RNN type of structure like this where like at any or at each time step the corresponding output $Y(t)$ is generated.

$$Y(t) = h_y(Z'(t))$$

And again the system state is which is now denoted by Z' that gets updated and. So, that I can generate the next output and so on ok.

$$Z'(t) = h_z(Z'(t-1), Y(t-1))$$

(Refer Slide Time: 12:52)



Sequential Learning and Inference

- The parameters of model need to be estimated from data
- HMM/LDS parameters: E-M algorithm or Markov Chain Monte Carlo
- RNN parameters: back-propagation through time (gradient descent)
- Repeated multiplication in long sequence -> vanishing/exploding gradients
- Solution: Long Short Term Memory (LSTM) networks

The slide features a blue header and footer. The background is white with faint icons of gears, a lightbulb, and a network. A video inset in the bottom right corner shows a man with glasses speaking. The NPTEL logo is in the bottom left corner.

So, now so, these are all models and each model comes with the different parameters. So, the question is how to estimate all those parameters from the data. So, in case of HMM or LDS they are these are probabilistic models. So, the parameters can be estimated using the E-M algorithm or Markov chain Monte Carlo methods. Especially for hidden Markov models there is the famous Baum-Welch algorithm for estimating the parameters that is based on the E-M algorithm.

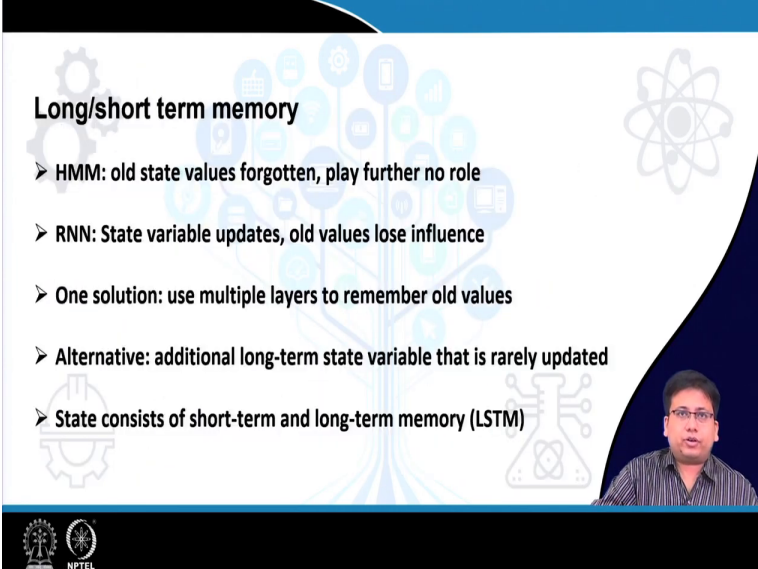
Now, in case of recurrent neural network parameters take that is basically all those matrices the like which we talked about the these matrices the A , B etcetera. So, they these matrices they have to be calculated in case of a an RNN and that can be achieved with the what is known as the back propagation through time with the help of gradient descent. So, just like in case of normal neural network all the edge weights they are estimated through the process of back propagation which is basically a gradient descent where we the derivatives are calculated and using the chain rule.

So, here also the same idea except that the back propagation now takes place through time and that is at every step we are considering the gradients of the previous time steps in and in that case also we are using the chain rule of differentiation as earlier; however, like when we are dealing with long sequences this gradient descent causes us to like multiply like a long sequences or these or long chains of these derivatives and as a result what happens is due to; Because we are

multiplying so many numbers like it often happens that these gradients they either tend to 0 or they tend to infinity that is I mean it like that is you can easily understand if the numbers are less than 1; then as you keep on multiplying many of them it will the product will be tending towards 0 and if they are greater than 1 and if you keep multiplying them it will be tending towards infinity.

So, this is these are the this is the famous vanishing or exploding gradient problem of RNNs and one way to solve it is the long short term memory networks of the LSTM.

(Refer Slide Time: 15:26)



Long/short term memory

- HMM: old state values forgotten, play further no role
- RNN: State variable updates, old values lose influence
- One solution: use multiple layers to remember old values
- Alternative: additional long-term state variable that is rarely updated
- State consists of short-term and long-term memory (LSTM)

The slide features a background with various icons related to technology and memory, including gears, a tree, a brain, and a network. A video inset in the bottom right corner shows a man speaking. The NPTEL logo is visible in the bottom left corner.

And now this the problem this LSTM network this solves our different problem also apart from this gradient problem. So, that is related to the memory of old values. So, in case of a hidden Markov model the value of Z the hidden value the latent state value that is sampled at every time step and once an old state value is forgotten it or is over written it does not play any further role in generating the outputs. So, at a like at any point the output Y that depends only on the current value of the system state.

But once that system state changes then that the old value does not play any further role in a hidden Markov model. Now in case of this recurrent neural networks the state variable keeps updating now the state variable is supposed to contain the or be some kind of a representation of

the past values that is we have already discussed that $Z(t)$ at any time t is in a sense it is a representation of all the inputs that have been received so far $X(1), X(2), \dots, X(t)$.

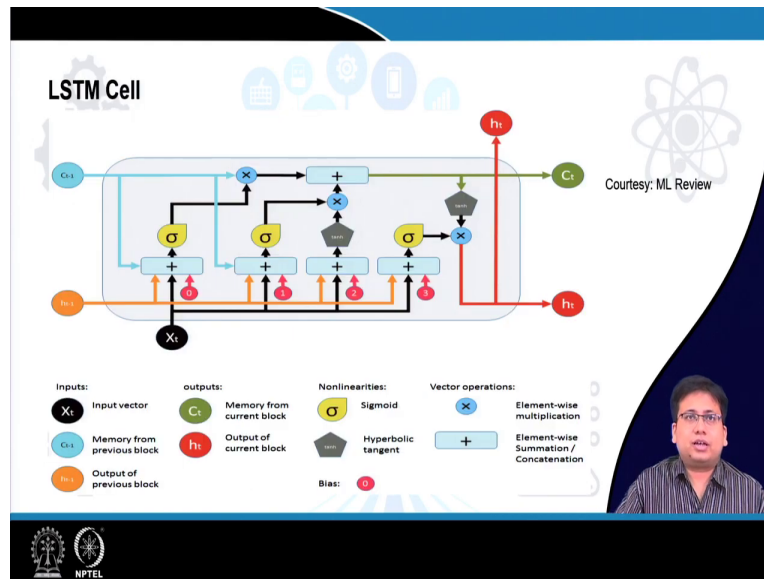
But as t increases the initial inputs like $X(1), X(2), X(3), \dots$ they become less and less important that is they are increasingly their effects are overwritten by the new values that are coming in. So, one solution to this problem might be to use multiple layers to remember the old values. So, we have already talked about instead of having a single state value like $Z(t)$ let us have multiple let us have a sequence of state values $Z(1), Z(2), Z(t), Z(k)$ and so on each of which is updated at every time step.

So, doing that might be a way to remember the old values. That is its basically we are increasing the memory of the network by adding layer of we can say we are adding a different memory cards or RAMs if you want to imagine it that way and. So, though all the different memory cards are stacked with each other. So, that we have more memory that is one way to do it.

The alternative is we can have a separate long term state variable that is rarely updated. So, in a sense it is the unlike the usual state variable $Z(t)$ which is updated at every on receipt of every input we can have a long term memory also which is carries the values of the old variables and it is not updated very frequently only when it the if the network somehow decides that now it can forget the old values and like focus on the new values only then we can then it can do that.

So, it is likes at every step it decides like whether some kind of a regime change has happened in the input. If that region change has happened then the previous values are not important. So, it can forget them and focus on the new region . But otherwise if it is continuing in the same region then it just holds that value in the memory.

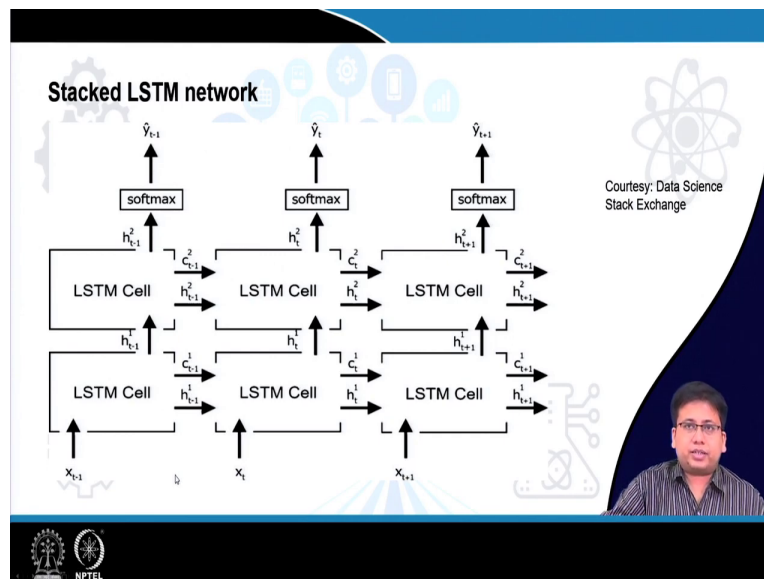
(Refer Slide Time: 18:50)



So, this is like what a LSTM or long short term memory cell looks like. So, like as you can see that there are different variables like c , h . This h is it is the hidden state or the or you can call the which is we can call it as the short term memory and then we have $c(t)$ which is like the cell state this is the long term memory this is rarely updated. In fact, there is a something called a forget signal which is generated at every whenever on the basis of the input X and if the forget signal if that is activated then only that is cell state gets updated with the current input otherwise it just carries on with the previous value which it had.

And the $h(t)$ on the other hand this is the short term memory which is updated at every step just like in case of a normal recurrent neural network.

(Refer Slide Time: 19:50)

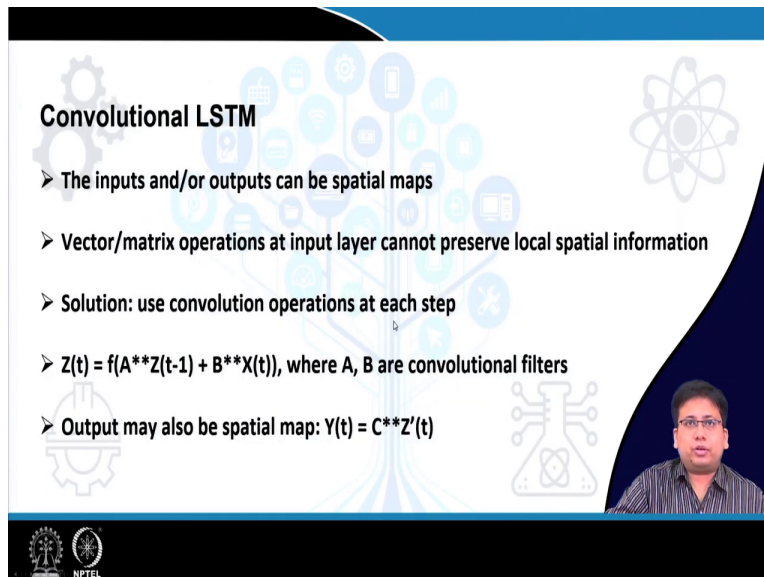


And now even in case of the these kind of LSTM, I may want to increase its memory by so, the like as I mentioned there is this long term memory called the cell state which preserves the old values, but then again it is the that cell state is also sometimes reset by like when that forget function is generated. So, then, but how to like how to remember those or how to preserve those old values if I need to need it for the final output right.

So, the way to achieve it is like just like in case of a RNN we talked about stacking multiple layers together $Z(1)$, $Z(2)$. Similarly we can also stack different LSTM cells etcetera. And so, each in each like we can have a one layer of LSTM cells on top of it we can stack another layer of these LSTM cells like this. So, like you can see that there are these horizontal operations that is in one layer the LSTMs like they function normally that is on receiving the input they update the cell state and the hidden state and then pass on to the next layer etcetera.

But that those hidden state $h(t)$ that is again passed on to the next layer of LSTM cells where also the similar operations takes place. And if there are the outputs they come out like at the end of these multiple layers.

(Refer Slide Time: 21:18)



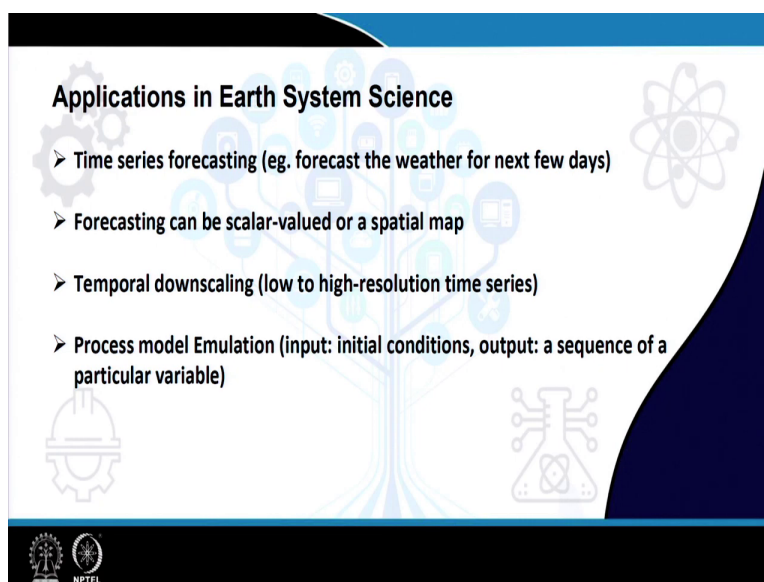
Convolutional LSTM

- The inputs and/or outputs can be spatial maps
- Vector/matrix operations at input layer cannot preserve local spatial information
- Solution: use convolution operations at each step
- $Z(t) = f(A * Z(t-1) + B * X(t))$, where A, B are convolutional filters
- Output may also be spatial map: $Y(t) = C * Z'(t)$

The slide features a background with various icons related to technology and science, including a gear, a lightbulb, a smartphone, a laptop, a network diagram, and a molecular structure. A small video feed of a presenter is visible in the bottom right corner. The NPTEL logo is at the bottom left.

So, just like we talked about the stacks RNN we can have stacks LSTM also and we can also have convolutional LSTM for the special situation where the inputs or the outputs they are spatial maps.

(Refer Slide Time: 21:34)



Applications in Earth System Science

- Time series forecasting (eg. forecast the weather for next few days)
- Forecasting can be scalar-valued or a spatial map
- Temporal downscaling (low to high-resolution time series)
- Process model Emulation (input: initial conditions, output: a sequence of a particular variable)

The slide features a background with various icons related to technology and science, including a gear, a lightbulb, a smartphone, a laptop, a network diagram, and a molecular structure. The NPTEL logo is at the bottom left.

So, if they like these inputs which we talked about if these inputs are instead of being say vectors or matrices if they are spatial maps. So, like we earlier we talked about the input operations being something like this right. So, like this is the system and this is the input. So, like there is some transformation matrices where this multiplication takes place.

The, but in the special case where this X these the input these are spatial maps then doing the usual vector or matrix multiplication may not be very useful because they will be they will not be able to preserve the local spatial information which is present in the like in that that kind of spatial data. So, as we had discussed in the last class the spatial these convolutional operations this is very important for spatial data because they define some kind of interactions among the neighboring pixels.

Now, the solution to this problem can be to use the convolutional operations at every step whenever we are receiving an input. So, we like we can like accordingly the input operations that can be defined in this kind of a way. So, in that case this the a system state Z that also we can express as like we can consider it as some kind of a spatial map like that and all like these A and B these can be considered to be convolution filters instead of transformation matrices and this is by this ** what I mean is the convolution operation.

So, instead of doing the usual matrix operation we can do the convolution operation and similarly when we are generating the output the $Y(t)$ that also can be the that also can be obtained by something another round of convolution on the system state. So, this way with what the kind of LSTM or RNN we get that is called a convolutional LSTM or RNN.

So, all the concepts discussed so far, they find lots of applications in earth system science. So, for say for example, in time series forecasting, say suppose we want to forecast the weather for the next few days. So, this is an example of a sequential prediction problem now this forecasting this can be either scalar valued or it can be a spatial map that is to say we may want to forecast either the total amount of rainfall being that is going to happen in the next few days or I may want a full spatial map of the rainfall distribution over a particular area for the next few days.

So, the if we are going for the second thing that is we want to predict a sequence of spatial maps then that becomes a sequential prediction problem for which we need something like a

convolutional LSTM. Another problem is that of temporal down scaling. So, that is where we have a low resolution time series may be each time step is like 1 day or something like that and we want to make infer a high resolution time series where the time steps are maybe say 1 hour or 6 hour something like that.

So, for that purpose also the concepts discussed earlier they have applications. And finally, in case of process model emulations. So, where we have set some initial conditions for a particular model that simulates some earth geoscientific phenomena, let us say some hydrological model or something like that and the expected output from that model is a sequence of a particular variable say the temperature or something.

So, the aim of this process model emulation is without actually running the simulation model we I will just predict its output based only on its initial conditions. So, the in the this is like again a sequence prediction problem for which we can use the concepts discussed earlier.

(Refer Slide Time: 25:44)



So, these are a set of references which shows the applications of these LSTM networks for the like in that various domains of earth system science. So, the first one this talks as you can see this is about the nowcasting of precipitation nowcasting means forecasting at short range as we talked about here. So, like another important application is that of temporal down scaling. So,

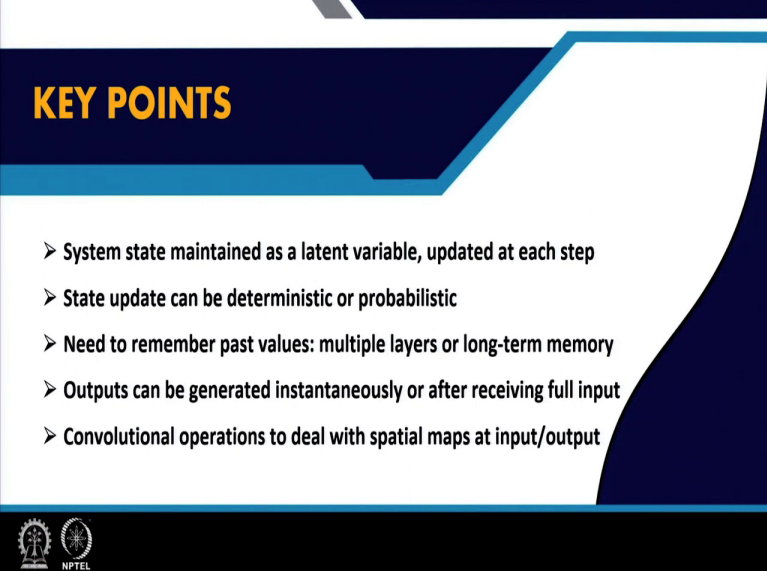
where we have? So, this is a an important interesting problem where the input is a time series and the output is also a time series.

So, this is the like the sequence to sequence mapping problem which we talked about. So, the input is typically it is a low resolution time series and the output is a high resolution time series. So, this temporal down scaling is often achieved using an RNN where we like where I have just mentioned like it can there can be a an encoder decoder framework where the encoder first receives the low dimensional sequence and then there is an a decoder which creates the output sequence.

And so, the convolutional LSTM which I mentioned this has been this was actually first proposed specifically for the purpose of rainfall forecasting from this from radar data. So, that the rainfall forecasting problem is actually cast as some kind of a spatio-temporal sequence forecasting problem we will study the LSTM the convo LSTM paper in greater details.


So, but that is a very important application where a famous machine learning technique was actually devised for the space for the specific task in earth system sciences and there are of course, many other applications in hydrology and so on.

(Refer Slide Time: 27:39)



KEY POINTS

- System state maintained as a latent variable, updated at each step
- State update can be deterministic or probabilistic
- Need to remember past values: multiple layers or long-term memory
- Outputs can be generated instantaneously or after receiving full input
- Convolutional operations to deal with spatial maps at input/output

 NPTEL

So, the key points to take away from this lecture is that like in these kinds of sequential problems the system state is maintained as a latent variable and it is updated at each state. The state update can be either deterministic or probabilistic. Now, and whenever we are like updating the step like basically we need to remember the past values which is usually achieved either through multiple layers of the state variable or as in case of a stacked RNN. Or in case of a or with the help of a long term memory unit as done in a LSTM.

And the outputs like they can be like either the single value or another sequence in case it is a sequence like in or in either case the outputs can be generated either instantaneously that is at every time step we can get an output or we can receive the full input and then only produce the output.

And especially if there are if the input is like spatial maps or if the outputs are spatial maps then we need convolutional operations at the input and output. So, for that we have like conv LSTMs or convolution cum recurrent networks. So, that brings us to the end of this lecture. So, we will like after in the next lecture 2 we will discuss some machine learning approaches which are like which are relevant to this domain of earth system science. So, till then goodbye.