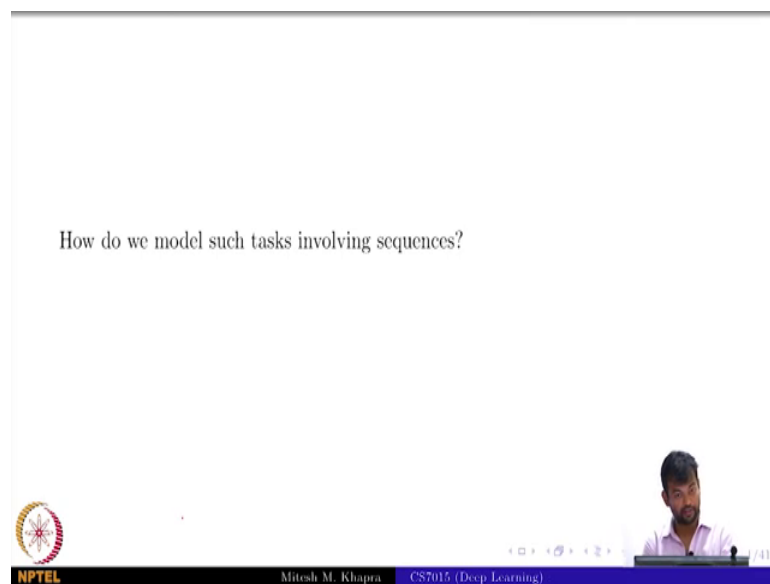


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 104
Recurrent Neural Networks

So, we have seen Sequence Learning Problems. Now, we are interested in the question of how to model these, right. So, we look at something known as recurrent neural networks.

(Refer Slide Time: 00:23)



And our question that we are interested is how do you model tasks which involves such sequences, ok.

(Refer Slide Time: 00:27)

Wishlist

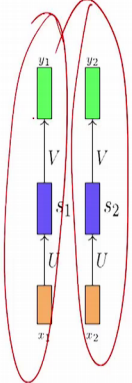
- Account for dependence between inputs
- Account for variable number of inputs
- Make sure that the function executed at each time step is the same
- We will focus on each of these to arrive at a model for dealing with sequences

NPTEL Mitesh M. Khapra CS7015 (Deep Learning) 2/41

So, here is the wishlist that we have. What will model will come up with should account for the dependence between inputs? Because that is the strong case that we have made, that the output actually depends on multiple inputs and not just a single input. You should also account for variable number of inputs because a video could be 300 seconds, it could be 20 seconds, 25 seconds; a sentence could be of arbitrary lines and so on.

And it also makes sure that the function at each time step is the same, right but every time step they are trying to do the same activity, ok. So, we will focus on each of these items from our wishlist and then try to arrive at a model for dealing with such problems.

(Refer Slide Time: 01:05)



- What is the function being executed at each time step ?
$$s_i = \sigma(Ux_i + b)$$
$$y_i = \sigma(Vs_i + c)$$
$$i = \text{timestep}$$
- Since we want the same function to be executed at each timestep we should share the same network (i.e., same parameters at each timestep)

Mitesh M. Khapra CS7015 (Deep Learning) 1/41

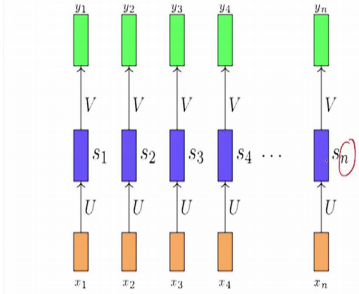
So, first let us ask this question what is the function being executed at each time step? What is the function being executed at each time step? Either should come after dealing. You have an input your ability to go blank on me is just amazing actually. You have a hidden representation and then you have an output. The first time we are seeing this situation in the entire course where we have an input a hidden representation and an output.

What is the function being executed? Remember the output is always a function of the input lecture 2 or 3 I do not know; definitely not lecture 14. So, what is the function? Can you write y_i as a function of x_i ? For my sake if not for god sake, you can, ok. What is it? Ok, first tell me what is s_1 ? $U s_1$, no nonlinearity, no bias which all that plus bias then no nonlinearity; who cares about nonlinearities, ok. And then what is y_1 ? Ok, some output function, ok. For some reason I have written sigma here. I will just call the output function as O always maybe in this case sigma would work but O is what I will call it just to make the this thing clear, ok; is that fine.

So, this is the function being executed at this every time step we can just write it using these two equations which are seeing for a first time, and i is a time step. Since we want the same function to be executed at each time step we should share the same network at every time step. That means, what do I mean by share the same network? Share the same parameters good. So, this is the same as this because U , V and b and c are the same, ok.

So, that is an easy way of taking care of the requirement that I want the same function to be executed at every time step.

(Refer Slide Time: 02:51)



- This parameter sharing also ensures that the network becomes agnostic to the length (size) of the input
- Since we are simply going to compute the same function (with same parameters) at each timestep, the number of timesteps doesn't matter
- We just create multiple copies of the network and execute them at each timestep

Mitesh M. Khapra CS7015 (Deep Learning)

And this parameter also sharing also ensures that the network becomes agnostic to the length of the input. Because now whether I have a word which has 10 characters or 20 characters it does not matter, because at every time step I am going to execute the same function. That is why it is important that at every time step we have the same function.

So, since we are going to complete the same function the number of times it does not matter and we can just create multiple copies of this network that we have and for any arbitrary length n we can still compute the output, ok. Still not quite there we still need to take care of a lot of things, but we are just slowly addressing each item from our wishlist.

(Refer Slide Time: 03:30)

- How do we account for dependence between inputs ?
- Let us first see an infeasible way of doing this
- At each timestep we will feed all the previous inputs to the network
- Is this okay ?
- No, it violates the other two items on our wishlist

Mitesh M. Khapra CS7015 (Deep Learning) /41

Now, how do we account for dependence between inputs? Or rather actually the, right way of asking this is how do we account for the case that the output actually depends on multiple inputs and not just the current input, ok? How do we account for that? Feed in the ok, good.

So, let us first see an infeasible way of doing this, ok. So, you are given the first time step x_1 , you have a network which predicts y_1 from x_1 . You know at the same second time step you also want to look at the previous inputs, so why not just feed it x_1 and y_1 and then try to predict y_2 . At the third time step feed in x_1, x_2, x_3 and predict y_3 and so on forever, is this fine probably the word infeasible is there. So, y_t is; so what is the problem with this? Good. So, I am looking in terms of the conditions that we have on the wishlist; which condition does it violate. What is the function being executed at each time step, ok? So, let us see.

(Refer Slide Time: 04:31)

- First, the function being computed at each time-step now is different
$$y_1 = f_1(x_1)$$
$$y_2 = f_2(x_1, x_2)$$
$$y_3 = f_3(x_1, x_2, x_3)$$
- The network is now sensitive to the length of the sequence
- For example a sequence of length 10 will require f_1, \dots, f_{10} whereas a sequence of length 100 will require f_1, \dots, f_{100}

Mitesh M. Khapra CS7015 (Deep Learning) 16/41

The function being executed at every time step is different. So, y_1 is function of x_1 , y_2 is a function of x_1, x_2 . Remember that this is not just saying that you are passing to inputs everything changes, because you now you need to have u_1 and u_2 here you need to have u_1, u_2, u_3 , right. So, everything changes; it is not the same function how many of you get this it is a different function being executed at every time step.

So, now, if I have a sequence of length 100 what happens? I need y_{100} which takes f_{x_1} to x_{100} as inputs. And has how many parameters? u_1 to u_{100} , right. You could you could share u_1 to u_{99} , for y_{99} and y_{100} but we still need those many of that, right. So, that network is now sensitive to the length of the input and on the length of the input goes you will have to construct more and more functions, right.

And imagine that if the training time the maximum sequence length that you had seen was 25. And suddenly a test time you get a sentence which is of length 30 you do not even know how to compute that because you have not train any parameters for doing that.

(Refer Slide Time: 05:36)

The diagram illustrates a recurrent neural network unrolled over time steps. It shows a sequence of inputs $x_1, x_2, x_3, x_4, \dots, x_n$ (orange boxes) at the bottom. Each input x_i is connected to a hidden state s_i (blue boxes) via a weight U . The hidden states are connected sequentially via a weight W . Each hidden state s_i is connected to an output y_i (green boxes) via a weight V . The network is shown for n time steps, with an ellipsis indicating intermediate steps.

- The solution is to add a recurrent connection in the network,
$$s_i = \sigma(Ux + Ws_{i-1} + b)$$
$$y_i = \sigma(Vs_i + c)$$
or
$$y_i = f(x_i, s_i, W, U, V)$$
- s_i is the state of the network at timestep i
- The parameters are W, U, V which are shared across timesteps
- The same network (and parameters) can be used to compute y_1, y_2, \dots, y_{10} or y_{100}

Mitesh M. Khapra CS7015 (Deep Learning) 17/41

So, then the final solution is actually to add a recurrent connection in the network. Why does this work? Ok, before that now can you tell me what is the function being executed at every time step? Assume there is a s_0 here; these are a s_1, s_2, s_3 up to s_n and there is a s_0 .

Now, what is the function being executed at every time step? Can you write it down if you? It would help if you think in terms of y_2 and not in terms of y_1 , y_1 is the boundary case was special case the thing in terms of y_2 or any other of the y 's. And first think of what s_2 is from s_2 y is straight forward. How many of you can write the function? So, s_i in general s_i is U into x_i plus W into s_{i-1} plus b . How many of you get this? And then what is y_i ? Again this has to be output functions, ok. But how does this solve our problem? Does it take care of everything on the wishlist? One the way we have written it in terms of i , which is the time step definitely the same function is getting executed at every time step. There is no doubt about that, right. Modulo this boundary case of s_1 where will assume that there is an s_0 , ok.

So, same function being executed at every time step. Can you deal with inputs of arbitrary length? Yes, as long as you ensure that the same function is executed its fine. Does it ensure that the output is actually dependent on the previous inputs? How?

Student: (Refer Time: 07:01), s_i .

Through s_i minus, right, so that is an interesting thing that this guy actually depends on this guy which depend on the previous input and also on this guy which in turn depends on the previous inputs. So, recursively you can see that you depend on all the previous inputs that you had, ok. That is a very neat way of ensuring that your output depends on all the previous inputs and you do not blow away the parameters blow of the parameters by sharing this recurrent connection, and that is this is a compact way of writing is that your y_i is now function of x_i s_i and has these parameters W , U , V and b and c . So, s_i is called the state of the network at times step i .

And what is c here? This is just for the sake of completion this is known as a recurrent neural network because of this recurrent connection and s_i is a state of the network at time step i . So, as when you start working in deep learning and you are dealing with sequence problems state of RNN or state of LSTM or state of GRV something that you will be hearing or reading often. So, this is what you mean by the state of the recurrent neural network this is the current state which kind of encode everything that is happened so far, right. It has a encoding of all the inputs that you had seen so.

The parameters of the network are W , U and V which I shared across time steps I obvious forget the biases. And the same network is getting executed every time steps I do not need to worry about whether I am computing y_1 , y_2 , y_3 or y_{100} , right. So, everyone agrees that this solution takes care of all the things that we had on our wishlist. How many of you agree with that?

(Refer Slide Time: 08:31)

- This can be represented more compactly

And this is a more compact way of representing that that you say that you compute s_i and then you are feeding it back. So, this is just more compact way of representing a recurrent neural network.

(Refer Slide Time: 08:41)

- Let us revisit the sequence learning problems that we saw earlier
- We now have recurrent connections between time steps which account for dependence between inputs

So, let us now revisit the sequence learning problems that we have seen. So, now, just correct each of these networks. So, what would happen? Each of these things I was thinking of all the inputs as independent. So now, what will I do? What is the only thing to be done? Just add the recurrent connection, right.

So, once you add the recurrent connection, now you can go back and relate to all these problems that one I am trying to predict the character which appears after e. I also have the information of d and e. And same argument you can make for all the other examples that you have. But I am trying to predict this final state I have the information of all the previous inputs here, ok.