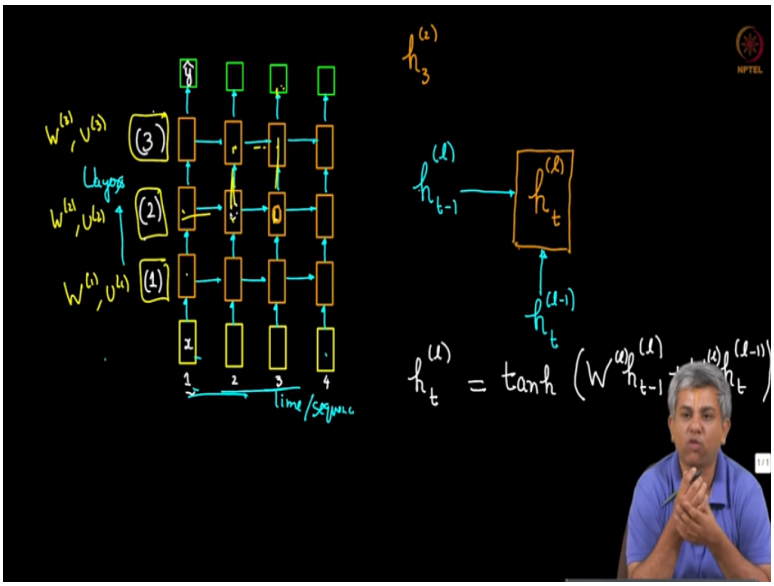
Machine Learning for Engineering and Science Application
Professor Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology Madras
Deep RNNs & Bi-RNNs

(Refer Slide Time: 00:15)



Welcome back in this video we will be looking at again at a very very low level at deep RNN as well as bi-directional RNNs the deep RNNs are particularly important in language especially Google translate for example uses deep RNN's at a certain level, so let me write that down so the Google translate that you will see if you go to get translate dot Google dot com we know because Google has published a paper that is uses at some level it uses RNN, now what are deep RNNs let us look at just one of these if I look at one of these within the RNN it is just an ANN as we saw with normal RNN's in a normal RNN all you had was one input layer one hidden layer and one output layer you know deep RNN all you do is that one single layer of the RNN actually become a deep neural network that is the only difference between a deep RNN and a normal RNN.

So now each of these could by themselves be LSTM etcetera so we are not going to discuss that but let us assume that each of this are actually deep networks now all that happens here is there is a connection between each layer or each time sequence remember in this direction we have time and in this direction we have layers decide how deep it is though sometimes you know by abuse

of notation even I have said this is depth this direction is not really depth, depth actually is along the layers, so this is either time or sequence etcetera, so now what is it you can think of this as if it is a single deep neural network unrolled, that is all it is unrolled and it is the same structure repeated again and again and of course the weight are also always the same , now what is the big deal about deep RNN.

So obviously there lately to deal with more complex structure, now of I look at some arbitrary input let us say this unit, so let us say this represent level one, this represents level two and this represents level three this of course if Y hat and of course this is time unit one , time unit two time unit three time unit four, so if I look at some element let me say this element, so let us draw that element now this is the one I am looking at right now, so this is H this hidden unit this is time sequences let us denote time sequence as P and by superscript let us give the level this is two or in general this is going to be L.

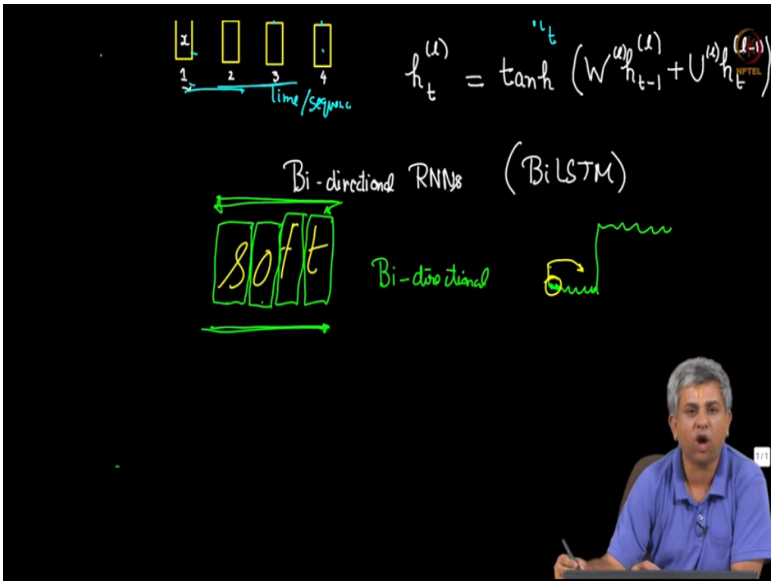
So in this specific case this element this element would be H three two, now what comes in is the previous one now as you can see time decrease here, so this is H t minus one at the same level and what come in below is instead of X which is what used to happen in a normal single hidden layer RNN in this case in a deep RNN what is below is actually etch itself T minus one so sorry T L minus one, so when this is the case we need to write the general expression remember our general expression would be HT L nothing much changes tan H of W times HT minus one L plus U times HT L minus one, now there is one small catch of course with each level.

So this one will have W1U1 this one will have W2U2 this one will have W3U3 that is the different here so in this case we would say that you have WL UL, so instead of one single W and U which is what we used for an RNN or a normal RNN you will have multiple W's and multiple U's that is the only different between a deep RNN and usual RNN now the other thing you can see is of course when you do back propagation through time or any back propagation it can get fairly complex because the gradient pathways are multiple to go from here to here you could go this way you could go this way.

You have all sorts of ways this is basically what tensor flow and things like that actually make a little bit easy they will draw a graph of what the dependencies are of each thing on everything else and they will automatically calculate the gradient for you so deep RNN are extremely useful

as I said in the beginning especially in language tasks but above and beyond already discuss there is other than computational complexity there no real notional complexity above and beyond what we have other than multiple W's and multiple U's.

(Refer Slide Time: 06:05)



The second thing that we look at is something called bi-directional RNN's sometimes you might even find the term by LSTM which simply is a bi-directional RNN using LSTM rather than the usual RNN this might be deep or not deep really does not matter. Now what are these four these are four tasks that actually are sequential but the sequence can go both backward and forward that is not only does the future depend on the past, so to speak but the past also depends on the future, now what would be an example of that let me give you a very-very simple example though you can think of several thing even in an engineering problem I will come back to that so suppose I write something of this sort and you have a optical character recognition tool which basically means this is handwritten and just like we saw with M nist you want to recognize a handwritten digit.

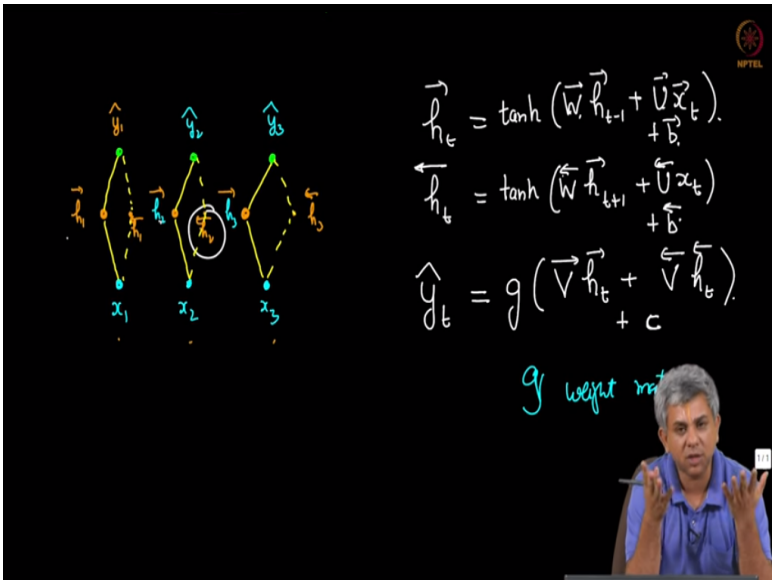
Similarly you want to recognize what is this word, now suppose the way usually RNN's will do it is this will be input one this image will be input two, this image will be input three and this image will be input four, now suppose I go only in one direction it will read S it will read O and it will not know whether this letter is T or whether it is F, so the probability of this will actually not be known because you are only seen that particular letter and the past letter, now however as

a human being if you see and identify this letter as T very clearly you can actually go back and correct yourself, in fact if I am not sure Microsoft editor equation editor uses that but you can actually see it if there is now and option in Microsoft equation editor called ink, where you can write things by hand it will actually go back and correct what it said before.

So if I actually read both backward and forward this is usually how we read even with our eyes he sort of guess what the middle letters are based on what happens at the end in such case you will need a bi-directional reading that is you will go this way read it you will go that way read it and sort of the joining of these two is what tells you what each letter is it is not only what happens in one sequence direction or the other sequence direction.

So you can in fact see this even in the sensor problem that I told you about suppose you want to guess whether a person sitting or not and you are at right at the beginning of a signal if you go back and look at that video you will see signals like this, but if I am right at the beginning of the signal how do I figure out what it is a part of at the beginning you actually have to guess by what happens in the future to see what the meaning of the first term is, so this is also an example even though we did not really use by Alice treat there but typically this is a good use case there too, now how do we actually do bi-directional LSTM it is a small tweak over the usual RNN.

(Refer Slide Time: 09:22)



So let us look at a figure I am going to reduce our boxes to circles here once again let us say this is X1 this is H1 and this is Y1 ha, so usually we would go in the forward direction and you would usually write something like HT vector is let us say tan H of W HT minus one vector plus Q XT vector, so this is what we usually do and plus of course one B vector also to add by a sometimes I feel forget writing the bias down, now what we are do when we are doing bi-directional LSTM's or RNN's IS add an additional vector remember X1 X2 X3 are fixed these are simply our inputs if you look at the soft word this would be S this ids O this is F and let us say if you have X4 that would be at T but I have a choice on what I can do for the hidden unit.

So not only do I add forward vector I add a reverse vector also and I will say HT in the reverse will draw the opposite vector is tan H, now H2 will not depend on H1 but it will actually depend on H3 the H2 reverse vector, so this I will say is W we will call this W forward call this W reverse even though these are not vectors you can add additional weights HT plus one vector plus U remains you add another correction vector here plus B inverse vector , so now you have added three new parameters this is just like what we did with LSTM UM's this is forward parameters even though these are not vectors I have called them forward just for you to see it you have inverse parameters now what about Y, if I look at Y T, YT will take an input not only from here but also from here so Y usually used to be simply some non-linearity of V times HT.

Now we are going to make it some V forward times HT plus V reverse X HT reverse plus of course actually I do not need to put inverse or forward let us simply say C so now you see here you have the bias unit you have these two vectors you have these two one another bias, so this is six parameters and nine parameters totally, so 9 so both in reading documents for getting out speech sometimes you can figure out what I am saying after I say a few words, so you can translation actually requires a bi-directional task you cannot translate a full sentence until you know the full sense of the sentence.

So you want to go back as well as you want to go forward as well as you want to go back in such cases bidirectional RNN ends are useful again over and beyond what we have said most of the other things are simply unrolling the graph and doing back prop otherwise there is no other difference from what we have done so far, so in this video we looked at both deep RNN's as well

as bi-directional RNN's these are just small tweaks depending on particular which use you want to put it to this depends on you these both are alternate sort of architectures for RNN's thank you.