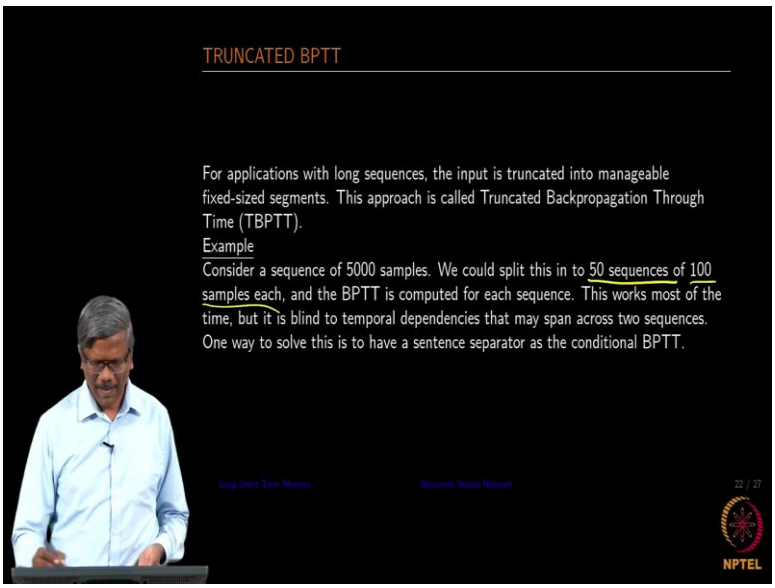
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture - 61
Truncated BPTT

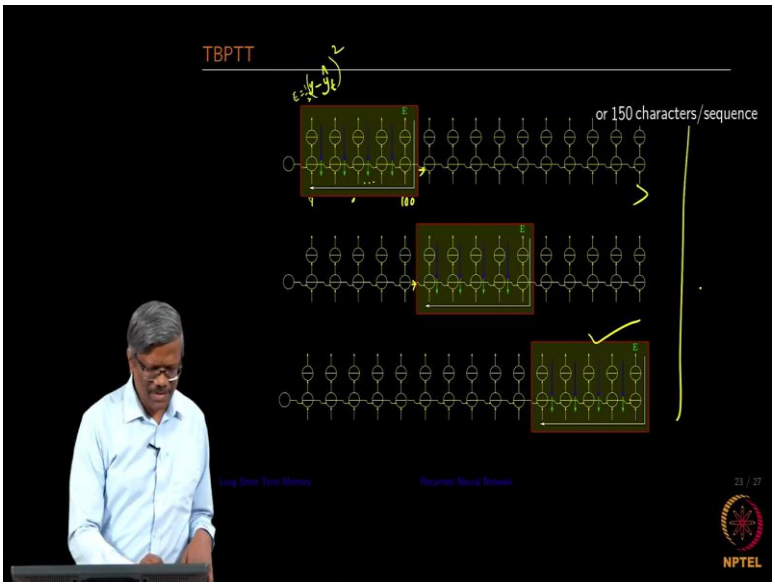
(Refer Slide Time: 00:15)



We mentioned that earlier LSTM could solve certain problems in the long term dependency and so on. Still, you cannot have a very large sequence for you to really do the backpropagation. So, for the convenience what we do is instead of having a 500-time slice, we actually create chunks ok. Let us say if we have about 500 samples in the time state that you want to process, we can split it into 50 sequences of 100 samples each, ok.

And then every chunk you do the forward pass backpropagation makes the correction and then go to the next chunk and then when you go to the next chunk, take the hidden state of the previous state, feed into this as your first input and so on alright. So, the input from the previous chunk is spread to the current chunk and you keep doing it until you finish all the chunks that you have created ok. So, this is called truncated backpropagation through time alright.

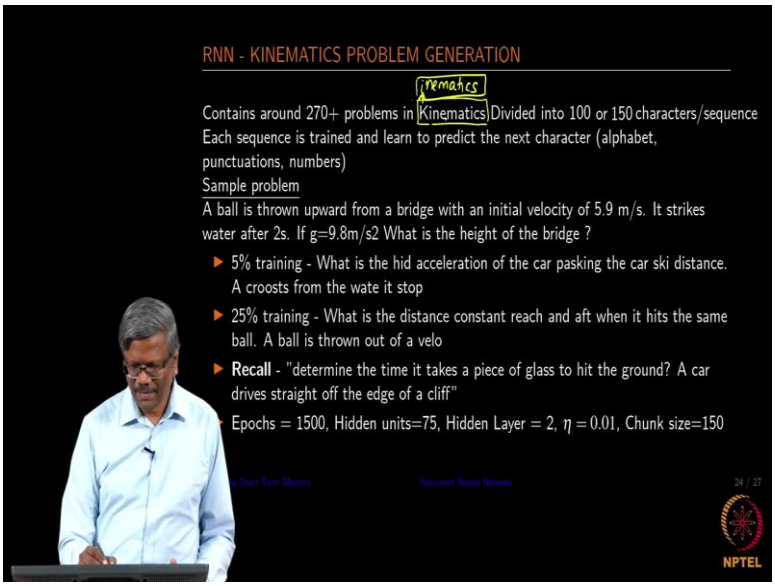
(Refer Slide Time: 01:41)



So, this is how it looks. For example, this is my sequence that I want to compute right. So, for example I have a long sentence so, instead of doing the backpropagation or forward pass to all the states and then do the backpropagation I take a small chunk. do the error calculation, do the backpropagation within this chunk initially and then ones it is computed, feed the value back to the next chunk here as the input.

So, the hidden values the previous hidden state of the previous chunk is fed as the input to the next chunk here. Do the computation for that chunk and then move on to the next chunk here and then do it ok. So, this is one way of doing the backpropagation. This is called truncate backpropagation through time ok.

(Refer Slide Time: 02:55)



Let us take one small example of how to really generate an English statement you know given a corpus. So, in this case what I have done is, I have taken a corpus that consists of 270 plus problems in kinematics and then I have divided this into 100 characters per sequence as I had shown earlier right. So, this sequence is my chunk now, so, every sequence contains about 100 characters.

The neural network knows nothing about this. All it knows is it contains characters like alphabets, punctuation, numbers and so on. It does not know the word, it does not know the spelling of any of those, it does not understand any of those initially. All it knows is there are so many characters and it has to process each character one at a time and then it knows that what is the next character to be predicted.

So, in this case what happens is suppose if you want to process this right and I am just taking one example here this is my input. So, my target would be like this, it will start with so, the first character would be k and it has to predict i. Then I should predict n, n should predict e and so on. So, this is how the characters are fed. So, in this case I have taken all the problems concatenated them into one single file and it is a sequence of characters.

So, from this I want to see whether the system is able to really generate useful statements or a useful problem statement this is the idea. So, it is a very difficult problem to solve right now so, and also, I do not have a very large corpus. I have only about 270 plus and

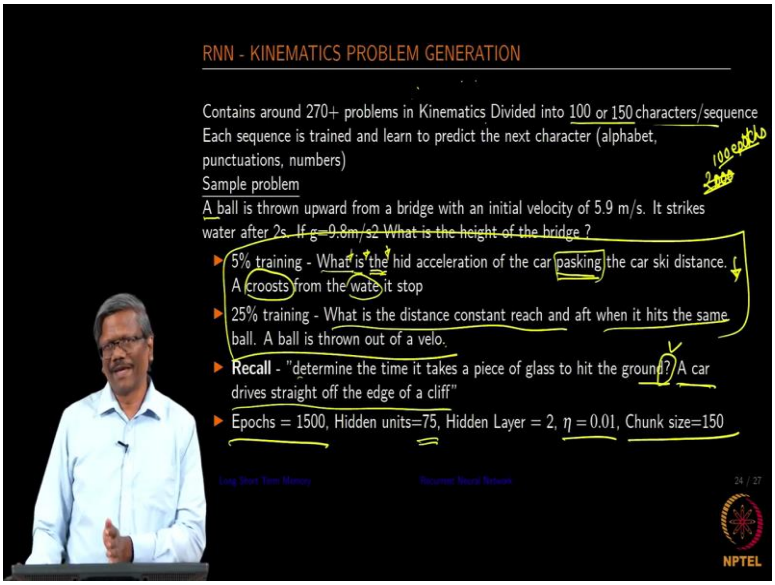
it consists of about 270000 characters max. It is very small in size. Here I am just inputting some problems of this type; a ball is thrown upward from a bridge with an initial velocity of 5.9 meters per second. It strikes the water after 2 seconds. If g is equal to 9.8 meters per second square what is the height of the bridge?

So, I have problems of this type and have 270 of them. I want the machine to read these problems and then try to predict what could be the next character. Try to predict when is the ending of a word and then finally, construct a sentence for me ok. So, this is the goal of that. I have one run, this is what I feed and then I feed about 100 characters per sequence.

So, here what happens is, so, every time when it reads it start reading from the first character. Let us say this is up to 1 to 100. It will read 100 characters and then for every character, it will predict the next one ok. So, since it knows what the input is it knows what is the goal standard that it should look at ok, so you get this here. So, every time when it predicts there is an error that happens.

So, it starts correcting it as and when it finishes one sequence and then at the end of some epochs so, it says that it is enough I have learned this then it will use the hidden layer state as the input to the next sequence here and then go. So, it does not know as I mentioned earlier any word the numerals or nothing. It all knows that everything is a character and it is not going to construct the sentence for me.

(Refer Slide Time: 07:39)



So, at the end of 5 percent of the training, I think I have got 2000 epochs. So, every 100 epochs I output some value ok. This you can really change. So, at the end of 5 percent of the training, after it has seen the entire sequence says it is able to construct words and also you see that spaces right, it is able to really predict what could be the next character after e. So, it has clearly found out the word and then sees hid acceleration and here packing, it does not learn cleanly what it is ok.

It is based on what is the predicted value it tries to construct a word and then here too. So, it is really funny to look at the sentence when it is really doing the training ok. At the end of 25 percent we can see some reasonable words coming out. Since I truncated at the end of 100 characters and 150, it does not show everything up.

So, this is only during the training process I keep you getting some values. At the end of the training you now say that give me a sentence starting with determining ok, it starts with determining, the time it takes a piece of glass to hit the ground ok. So, probably it has seen somewhere that either the ground ended with a question mark several times so, the pattern has been captured and then it ends with the question and then a car drives straight off the edge of the cliff of a cliff so, it is interesting there too.

So, this is the recall, you can see that you know this there are not many spelling mistakes even though the sentence is not semantically right ok. You can see it is trying to really create a sentence that is syntactically right ok. So, I am really baffled by the question mark also it adds at the end you know. So, with rnn you know this is the possibility, you can do all of these with the run.

So, in this case I have used about 1500, not 2000 and then I have the hidden units as 75 and then I have two layers there ok. So, I have when you unwrap, there is one layer and there is one more on top of it and they are all connected like this and also vertically up.

And then I used the learning parameter as 0.01 and my chunk size is about 150. This is what I had ok. So, this is one example. So, this is a very simple example through which I had shown that it is possible for you to really construct a sequence of characters that make some sense using an.