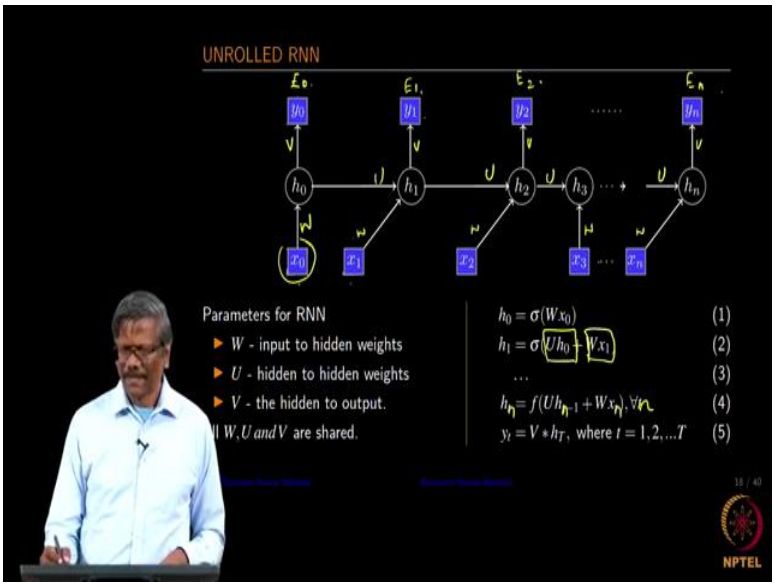
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture - 55
Unrolled RNN

(Refer Slide Time: 00:15)



So, if you look at this, we are starting from here there is an initial state given and then using the initial tray state and the initial input value, we compute the activation and then using v we compute the output. So, here we have W this is I am sorry this is U and this is W. So, we; so, each one if you take it apart each one is a small backpropagation network but they are connected through the hidden layers through time right.

So; that means, every time when you create a new input a new value you get an output here. So, there is an error computed at here; that means, error values are spread across different time slice as well in this particular model. So, in this is an unrolled one where you have n time slices and then there are n inputs that you feed in the time slice one after the other.

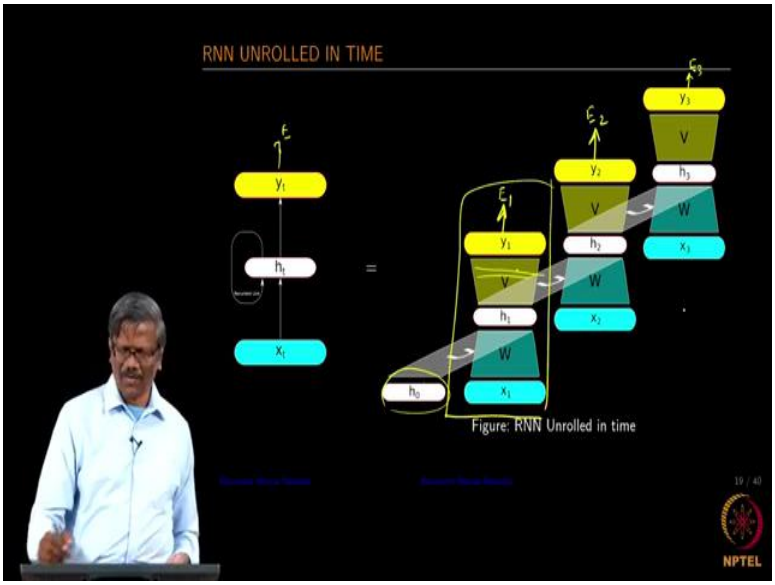
$$h_0 = \sigma(wx_0)$$

$$h_{1=}\sigma(uh_0 + wx_{1)}$$

And then for every slice time slice, you have an output layer or output generation for every layer I am sorry every time slice there is an error generation. So, also we need to consider all of this when we do the backpropagation through time ok. So, please understand this part right now that we can unroll a recurrent neural network through time and we can share the values of the hidden weights of through time.

And we can compute the output in the same manner that we had computed the output earlier. So, only change that we will see is the computation of the h_1 or the h using the previous value that we have stored in the memory and then the current input and the weight vector connecting the hidden layer right. So, this is clear? Ok.

(Refer Slide Time: 03:48)



Let us move on to the next one again different representations I just want to make you comfortable with various representations that you will see in the technical papers ok.

So, this is one way of writing it and then this is another way which gives you in the matrix form ok. So, you can see that the h 1 starts here and then h 1 is computed using h naught u and h 1 W and then y 1 is computed using the context vectors that you find here and the activation function that we computed using h 1 and W and the previous state and so, on ok. So, each one is a separate about propagation neural by itself.

So, you can compute the error in each of these and then you propagate the error through time to update all the weights.