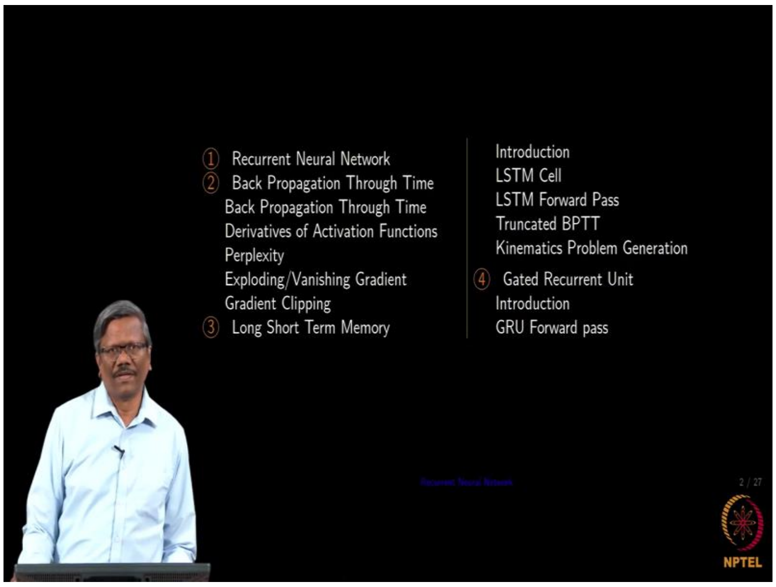
Applied Natural Language Processing

Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture – 57
BPTT – Forward Pass

In the last session we saw the architecture of the recurrent neural network, we saw some small examples of how we can use RNN to solve certain problems in the natural language processing. Today we will look a little deeper into the RNN and see how it works and so on all right ok.

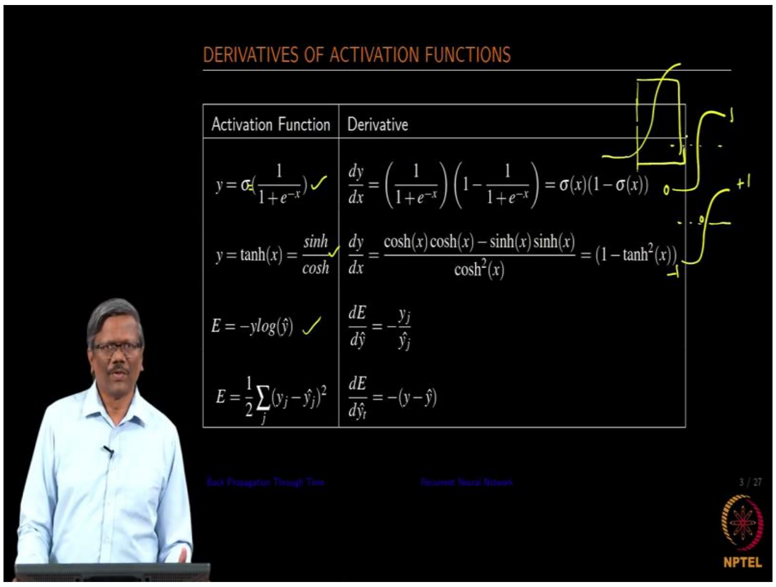
(Refer Slide Time: 00:39)



So, in this class, I will be talking about backpropagation through time if only call it as BPTT and then we will look at the derivatives and then see how we can backpropagate the errors and then train the network in the recurrent neural network and then we also look at what is meant by perplexity. Later we will look at exploding and vanishing gradient, we also look at a small technique in terms of clipping the gradient to solve the exploding gradient problem.

Then later we introduced a new mechanism by which we can solve certain problems exposed by recurrent neural networks, we call it a LSTM or Long Short Term Memory cell. We will see the forward pass and then how we can use LSTM to address some problems that we face in the RNN. We also take one small example in terms of generating a sequence through LSTM and then later we look at another modification of the RNN called gated recurrent unit. We will see the diagram of that and then see how it works and then close this session right.

(Refer Slide Time: 02:05)



DERIVATIVES OF ACTIVATION FUNCTIONS

Activation Function	Derivative
$y = \sigma(\frac{1}{1+e^{-x}})$ ✓	$\frac{dy}{dx} = \left(\frac{1}{1+e^{-x}}\right)\left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x))$
$y = \tanh(x) = \frac{sinh}{cosh}$ ✓	$\frac{dy}{dx} = \frac{\cosh(x)\cosh(x) - \sinh(x)\sinh(x)}{\cosh^2(x)} = (1-\tanh^2(x))$
$E = -ylog(\hat{y})$ ✓	$\frac{dE}{d\hat{y}} = -\frac{y_j}{\hat{y}_j}$
$E = \frac{1}{2}\sum_j (y_j - \hat{y}_j)^2$	$\frac{dE}{d\hat{y}_t} = -(y-\hat{y})$

Probably I have to read lot more to understand this rather than just listening to the listening to this lecture also go on then take a look at recent papers and their papers published in this particular topic to see how they really are addressing, the vanishing gradient problem in the backpropagation through time and so on.

So, we will just look at some of the important things that you would see in all the neural networks and it is going to be refreshing this again to you. We are going to be looking at some activation function and their derivatives it is right. So, we use the activation function to control the values of the hidden layers, output layer values, and so on you remember that. And then we also take the derivative when you want to do the backpropagation we take the derivative and then take the values back and forth so, that we can train a neural network.

So, we have seen a few activation functions; one is a sigmoid right and then we also saw tan h or hyperbolic tangent. We were looking at 2 different mechanisms of finding errors and then trying to use that to reduce the error during the backpropagation; one is using the cross-entropy and then the second one is the mean square here right.

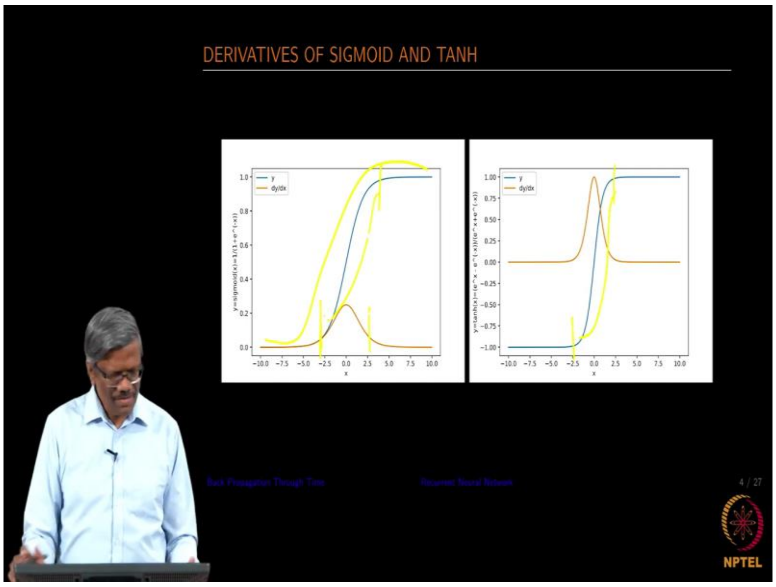
So, all of them should have the derivative so, that you should be able to continuously do the backpropagation in order to train a network. So, why are these functions chosen? These functions are continuous in nature and they are differentiable. So, if you do not have a differentiable function you cannot use it as an activation function. So, that is the reason why we choose some of these functions as the activation function.

So, for example the sigmoid is equal to $\left[\frac{1}{1+e^{-x}}\right]$

So, if you want to take a derivative of this it is equal $\sigma(x)(1-\sigma(x))$ ok. And then if you want to differentiate the hyperbolic tangent function it gives you 1 minus tan square hit gives you 1 minus hyperbolic tan square x right so, it is continuously differentiable.

If you take the sigmoid it goes from 0 to 1 if you look at the hyperbolic tangent it is from -1 to 1 right. So, you have so, I have to draw it cleanly here it should be like this right. So, you have a good amount of space where you can differentiate it cleanly right in both these functions all right. So, that is the reason why we look at the activation function which are differentiable, it is very important to have a continuous function for you to use in the activation for you to use it as an activation function.

(Refer Slide Time: 05:35)



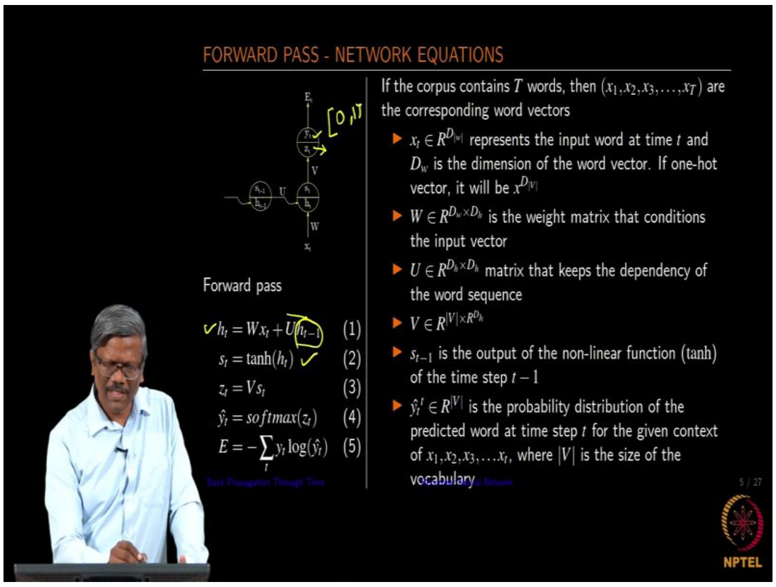
DERIVATIVES OF SIGMOID AND TANH

y
dy/dx
y=sigmoid(x)=1/(1+e^(-x))
y=tanh(x)=(e^x - e^(-x))/(e^x+e^(-x))
x
NPTEL
4 / 27

Then let us look at the derivatives of this you know we saw it as a formula let us look at it through the graph then you will understand it even more better. We have this sigmoid right like this and then we have the derivative in red.

So, you can see that if you see the derivatives are available in these regions for you and then if you look at the hyperbolic tangent function you have the derivatives again in this region. So, ideally if you have all the values in these places you are going to have a derivative for backpropagation and so on ok. So, we aim to keep our values in these places ok all right.

(Refer Slide Time: 06:37)



FORWARD PASS - NETWORK EQUATIONS

Forward pass

$$h_t = Wx_t + U h_{t-1} \quad (1)$$
$$s_t = \tanh(h_t) \quad (2)$$
$$z_t = V s_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4)$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

If the corpus contains T words, then $(x_1, x_2, x_3, \ldots, x_T)$ are the corresponding word vectors

- $x_t \in R^{D_{|w|}}$ represents the input word at time t and D_w is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t^t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

5 / 27

NPTEL

So, once we have a good continuous function we can use it for our forward pass to really change the value between either -1 to +1 or 0 to 1 depending on whether you choose tan hyperbolic tangent function or sigmoid function ok.

So, now let us look at the forward pass in the recurrent neural network. So, I have given a very small and simple model for you to understand. So, we have the input coming in as x_t at time slot t and then the input values are conditioned by the matrix w that connects the input and the hidden units, and then we have the hidden units getting the value by using the dot product of W and x_t.

And then we have an s_t that represents the state of this neuron hidden neuron which is computed using the x_t and then we have a matrix connecting or the weight connecting the state of the hidden neuron on the output neuron ok. So, we have a net value computed here that is called z_t and then we do a softmax and then get our classes right and then later we find the error between the value that we computed and the gold value that we have added as the standard or the target output ok.

So, the difference is computed using sthem that and the gold value that we have set earlier ok. So, now, look at the equation that we have so, let me take off these. So, h t is computed using W x t and U is the one that connects the previously hidden layer and the current layer right. So, or we can say it as a previous state and the current state. So, this one represents the previous state of the hidden unit and then this represents the current state of the hidden unit.

So, we have some value that is coming from the previous time slides that we have to incorporate into this. So, how are you going to do that? First we will find the state of the current one ok. First what we do is we find the current state of h t using the input and the W and then sum it with the U the one that is connecting the previous state of the hidden neuron and the current state and taking the previous state value of the neuron hidden neuron right.

So, by doing that you get the value of h here so, we compute the value of the current state of the hidden neuron using these 2 right ok. So, what next? So, once we have computed the current state of the hidden neuron. So, we use hyperbolic tangent function to get the state of it ok. This state is translated between minus 1 and plus 1 using a hyperbolic tangent function here ok.

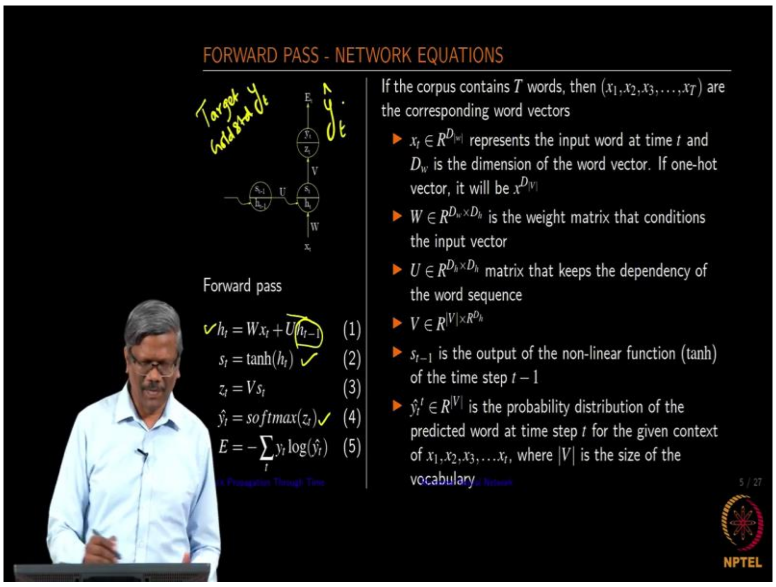
So, in all these you should also you know not just think in terms of the variables here, but look at in terms of how the matrix or vectors look like when you move from the input to the h t and then what happens in the state and then when you multiply with the vector with the matrix again to get the z t you know you just look at it or visualize it in terms of the matrix in your mind when you do that.

And then also when you use an activation function there you know that the values are mapped onto the values between minus 1 and plus 1 if use at tang hyperbolic tangent or they are mapped between 0 and 1 if you use a sigmoid. So, it is nothing more than the set of values between 0 and 1 or minus 1 and plus 1 ok. So, you just keep that in mind whenever you do this.

And then we compute the value of z that is the state of the net output time slice t using the weights connecting the state of the hidden unit here and the output neuron right. So, now, we compute E z t using V and s there ok. So, we got that and then later we obtained the computed output using softmax. So, what softmax does is again it maps the value that you obtain in the z t between 0 and 1 right. So, it also distributes the value in such a way that the sum of all the values would be equal to 1.

So, you can say that this is also doing the job of a sigmoid write the values are going to be between 0 and 1 in this case too, but with the only one condition all the values when you sum up they become equal to 1. So, that is the only condition otherwise it acts like a sigmoid function all right. So, once the output is computed we now have to find the error right.

(Refer Slide Time: 13:31)



FORWARD PASS - NETWORK EQUATIONS

Forward pass

$$h_t = Wx_t + U\boxed{h_{t-1}} \quad (1) ✓$$
$$s_t = \tanh(h_t) \quad (2) ✓$$
$$z_t = Vs_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4) ✓$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

If the corpus contains T words, then $(x_1, x_2, x_3, \ldots, x_T)$ are the corresponding word vectors

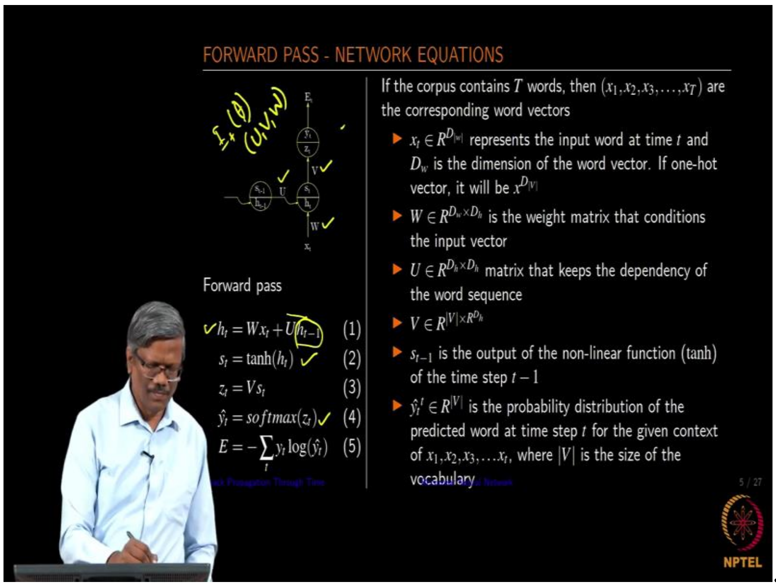
- $x_t \in R^{D_{|w|}}$ represents the input word at time t and D_w is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t^t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

5 / 27

So, we know the target so, we also call it as the gold standard which would be let say y t and we have y hat t now and we need to find the error and then minimizing this error is going to really give you the equilibrium state, you know when you start minimizing it and then when the value no longer changes you understand that the network has reached the state of equilibrium. That is the very ideal situation, but we stop at some point usually you know when the value between the previous state error and the current state error is equal to a very small value we stop that. So, that is the way we compute the error here.

So, once we compute the error now we have to push the value back through the network. So, that the error is learnt the weights the error is learnt what is the meaning of that when we say we are learning the values of V W and U are adjusted, those are the parameters that we are going to be looking at in terms of reducing the error of the entire network.

(Refer Slide Time: 15:01)



FORWARD PASS - NETWORK EQUATIONS

Forward pass

$$h_t = Wx_t + Uh_{t-1} \quad (1)$$
$$s_t = \tanh(h_t) \quad (2)$$
$$z_t = Vs_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4)$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

If the corpus contains T words, then $(x_1, x_2, x_3, \ldots, x_T)$ are the corresponding word vectors

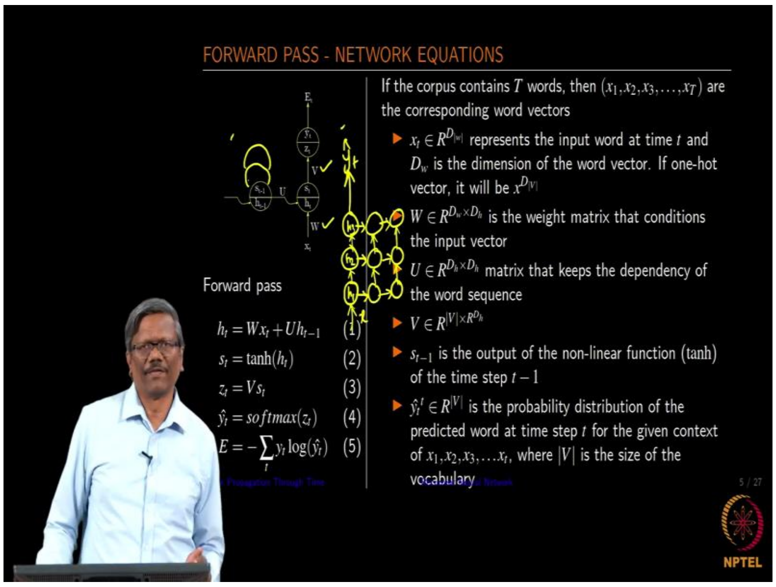
- $x_t \in R^{D_{|w|}}$ represents the input word at time t and D_w is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t^t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

5 / 27

.

So, when we say E_t θ; that means, it includes U, V and W. So, this is for only one small set ok. So, here we have considered the neurons in the hidden layer in the horizontal fashion when you unroll it right.

(Refer Slide Time: 15:27)



FORWARD PASS - NETWORK EQUATIONS

Forward pass

$$h_t = Wx_t + Uh_{t-1} \quad (1)$$
$$s_t = \tanh(h_t) \quad (2)$$
$$z_t = Vs_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4)$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

If the corpus contains T words, then $(x_1, x_2, x_3, \ldots, x_T)$ are the corresponding word vectors

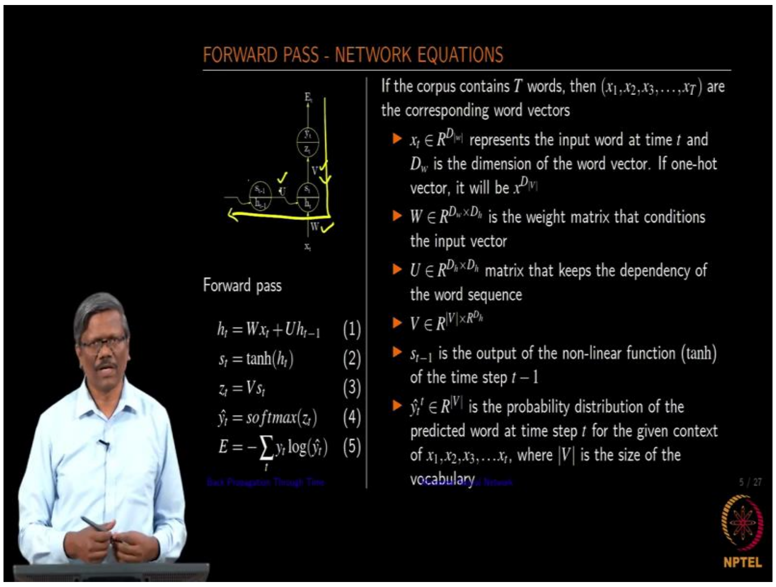
- $x_t \in R^{D_{|w|}}$ represents the input word at time t and D_w is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t^t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

5 / 27

It is also possible that you can have neurons stack one above the other, meaning there are multiple layers between the input and the output. So, here there is only 1 hidden layer I can have more than 1. So, and then I can x through this. So, when you create something like this you have when you unroll you are going to be like ok. So, this is how you can actually add more layers to that network ok. So, what is the advantage of adding more layers to the neural network?

Remember we showed that in the x R problem by just adding one more neuron in the hidden layer. So, we are able to solve the non-linear problem using 2 neurons a variable to solve the x R problem by adding one additional neuron in the hidden layer unit. So, by adding more and more so, you are actually increasing the capability of the neural network to solve problems which are a lot effort to address using the normal operations ok. And let us see how we get to that place ok. So, this is one simple network where we are able to compute the error using the forward pass here.

(Refer Slide Time: 17:27)



FORWARD PASS - NETWORK EQUATIONS

Forward pass

$$h_t = Wx_t + Uh_{t-1} \quad (1)$$
$$s_t = \tanh(h_t) \quad (2)$$
$$z_t = Vs_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4)$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

If the corpus contains T words, then $(x_1, x_2, x_3, \ldots, x_T)$ are the corresponding word vectors

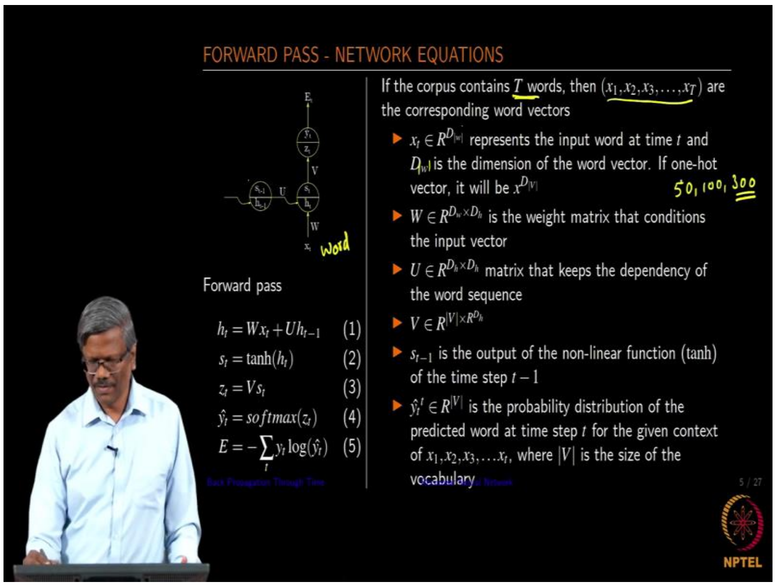
- $x_t \in R^{D_{|w|}}$ represents the input word at time t and D_w is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t^t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

5 / 27

So, then we have to do the backpropagation we have to find the value by which we want to update the V, we want to find the value by which we want to update the W and then we want to find out the value or some delta value using check and update this. And then you are not only having this you have more of this on your side right and then you have to take those values across as well when you make the updation in each of these weights.

Let us look at first the complexities of this and then in terms of the size and then move on to the backpropagation part ok. So, let us assume that we have t words in the vocabulary; that means, we are going to be providing the words one word at a time and there going to be T words it is the total number of words in the corpus I am sorry not in the vocabulary ok. So, that will be given one at a time. So, we have t words in that, X_t represents the input word at a time t and D w that we have here is the dimension of the word vector suppose you are inputting the x t in terms of the word right.

(Refer Slide Time: 18:47)



FORWARD PASS - NETWORK EQUATIONS

If the corpus contains $\underline{T}$ words, then $\underline{(x_1, x_2, x_3, \ldots, x_T)}$ are the corresponding word vectors

- $x_t \in R^{D_{|w|}}$ represents the input word at time t and $D_{|w|}$ is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$ 50, 100, $\underline{\underline{300}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t^t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

Forward pass

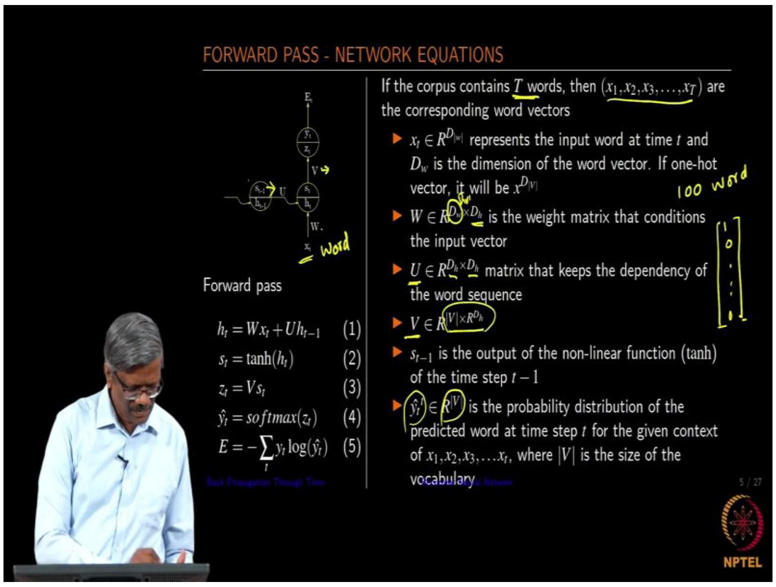
$$h_t = Wx_t + Uh_{t-1} \quad (1)$$
$$s_t = \tanh(h_t) \quad (2)$$
$$z_t = Vs_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4)$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

5 / 27

NPTEL

There are 2 ways you can do it, one is you can actually input the word vector. So, when you input the word vector the vector will have some elements. So, usually as I mentioned earlier you can take 50, 100, or 300. So, the dimension of this would be 300 in some cases or 50 in some cases and so on depending on what you want to do all right. And then if you are considering a 1 hot vector then this size of x_t will be equal to the vocabulary size right. Why is it so?

(Refer Slide Time: 19:33)



FORWARD PASS - NETWORK EQUATIONS

Forward pass

$$h_t = Wx_t + Uh_{t-1} \quad (1)$$
$$s_t = \tanh(h_t) \quad (2)$$
$$z_t = Vs_t \quad (3)$$
$$\hat{y}_t = softmax(z_t) \quad (4)$$
$$E = -\sum_t y_t \log(\hat{y}_t) \quad (5)$$

If the corpus contains T words, then $(x_1, x_2, x_3, \ldots, x_T)$ are the corresponding word vectors

- $x_t \in R^{D_{|w|}}$ represents the input word at time t and D_w is the dimension of the word vector. If one-hot vector, it will be $x^{D_{|V|}}$
- $W \in R^{D_w \times D_h}$ is the weight matrix that conditions the input vector
- $U \in R^{D_h \times D_h}$ matrix that keeps the dependency of the word sequence
- $V \in R^{|V| \times R^{D_h}}$
- s_{t-1} is the output of the non-linear function (tanh) of the time step $t-1$
- $\hat{y}_t \in R^{|V|}$ is the probability distribution of the predicted word at time step t for the given context of $x_1, x_2, x_3, \ldots x_t$, where $|V|$ is the size of the vocabulary

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

5 / 27

NPTEL

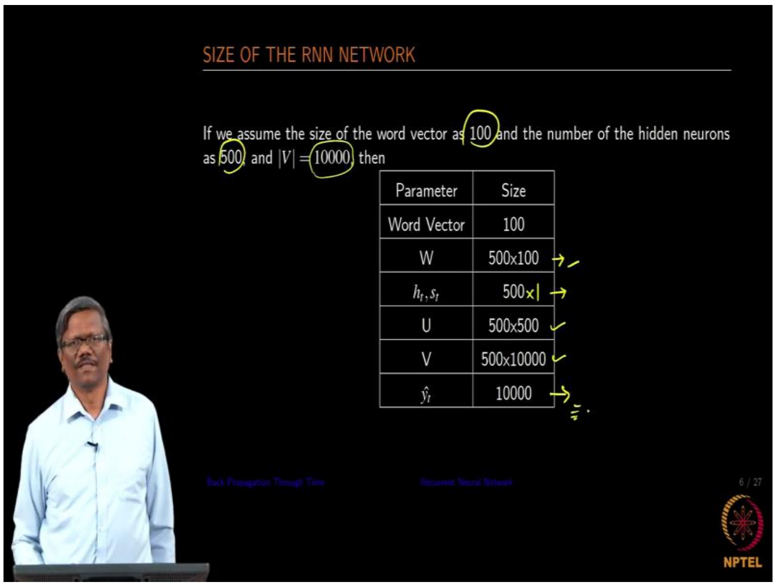
If you are considering 100 word vocabulary, so each word will be represented by the index right, and then for the first word number 1 will be 1 rest will be 0. So, we will have 100 elements in that so, this particular value will be equal to D V if we use 1 hot vector ok. So, keep that in mind all right.

So, next is the one that connects the hidden unit and the input vector. So, the size would be. So, we have decided the size either it will be D w or D v and h is the number of neurons in the hidden layer will be given by this number. So, this is a matrix that contains D w into D h elements right and all other them are in the real space and then U is another matrix that is coming in from the hidden unit to the hidden unit.

So, which will be a square matrix, which will be of size D h $_x$ D h. Then v is another matrix that connects the state of the hidden unit and the output neuron and that will be equal to V ÷ R × D_h. So, since we are connecting it from the hidden layer the size will be that one of the sizes will be D_h the other one will be equal to the vocabulary size right.

So, since we are using the softmax so, it is going to be spreading the entire probability across the words right so; that means, we are going to have the size of V there. So, this is what we have, and then s t minus 1 is the output of the non-linear function that comes from the previous state. The output that we have here will be of size R is this clear. So, this is the size that we are talking about the values of a V, W and U are shared across. So, you will have only 1 U, 1 V and 1 W throughout the implementation all right.

(Refer Slide Time: 22:19)



SIZE OF THE RNN NETWORK

If we assume the size of the word vector as 100 and the number of the hidden neurons as 500 and $|V| = 10000$ then

Parameter	Size
Word Vector	100
W	500x100
h_t, s_t	500x1
U	500x500
V	500x10000
$\hat{y}_t$	10000

Back Propagation Through Time Recurrent Neural Network 6 / 27

So, give you an idea of how big it will be let us take some small example, if you assume the size of the word vector to be 100 and the number of hidden neurons as 500 and the vocabulary size is 10,000 ok. So, the weights connecting the input and the hidden units is going to be of this size 500 into 100 and then this size of the or the state size would be equal to 500 into 1 and then U would be 500 by 500 and then V would be 500 into 10000 and the yt value would be 10000.

So, this is only for 1 unit that we are talking about this right. So, when you do the when you unroll this you will have several of y‘s correct for every time slot there will be any t. This would not change, this would not change these things would not change, but there will be a change in this.