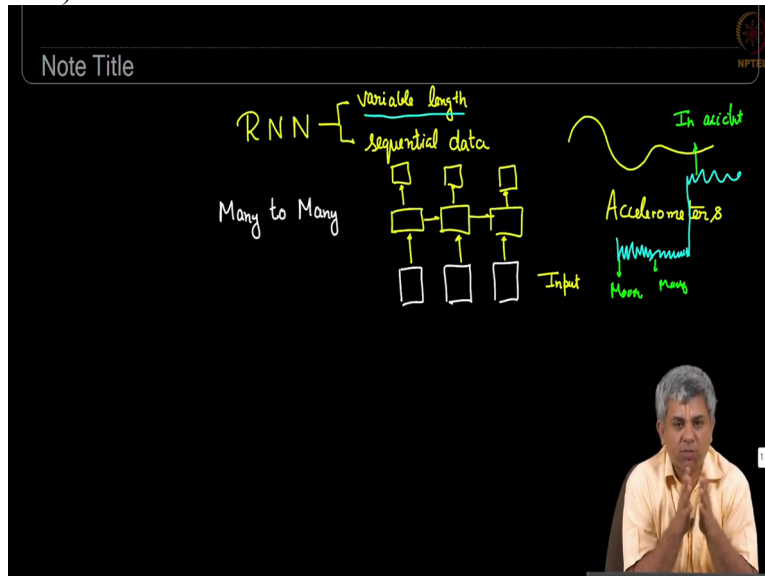


Machine Learning for Engineering and Science Applications
Dr. Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology Madras
Example - Sequence to Sequence Classification Example

(Refer Slide Time 0:15)



Welcome back. In the last video, we saw an introduction to RNNs. RNNs remember stands for Recurrent neural networks. I told you about 2 important properties that RNNs have that typically CNNs and ANNs cannot handle. One of them was variable length, the 2nd thing is sequential data. So as I had had a few times in the last video, the basic purpose of RNNs is to be able to handle variable length sequential data. In this video, I will show you a short example, again borrowed from MATLAB's examples, this time explicitly borrowed from what MATLAB has given in their deep learning toolbox.

And hopefully, you will see what specifically we mean by variable length. Sequential data should already be intuitive to you. So the purpose of this video is not to actually introduce you complete to RNN but to introduce you to what the use cases, you know how are we going to use it. okay. Hopefully it will give you some intuition about what we are going to do. Now when we were looking at RNNs, we looked at various classifications. These classifications are not really official classifications but they are given by a very good researcher, Andre Karpathy.

And what we are going to look at in this example is a many to many classification. So we call that a many to many classification, there were 2 subsections that we looked at, again this is Karpathy's way of doing it. So suppose you have various inputs, they could go through the sequence of hidden layers and suppose you have an output corresponding to each of these inputs. This would be a many to many classification. Now we will do one example of that sort shortly within this video.

Now I would like to draw an analogy between what you will see here and this is called sequence to sequence classification. So I am going to give an analogy between this and what happened when you did something like semantic segmentation when we were looking at CNNs. In CNNs, we try to label each pixel as belonging to body 1, body 2, or body 3 as Dr Venapathy showed you. So a sequence to sequence classification is somewhat similar. Okay.

What does that mean? It means that suppose you have a time sequence and a few possible events could happen, okay. So a few possible events, so I will give you one good example of this. So let us say you are driving a car and you have your cellphone in there. Remember that cellphones have accelerometers. If you do not know this, it is an interesting thing. So what a cellphone has within it is some way to figure out which way the acceleration is going and the reason for this is, a simple reason for this is so that when you rotate your cellphone, the picture can rotate too.

So if you have a smart phone, it has, first brought out by iPhone as you remember, so this have this capability to rotate your picture because it needs to know which way you are accelerating and which way is up and which way is sideways. So cellphones have these accelerometers. So it is a you keep a cellphone within your car okay a smart phone within your car and by your acceleration signal, it should be able to say whether you are stationary, whether you are moving or at a you know whether you are accelerating or whether you are in an accident, okay.

Now this can be particularly useful and I am not sure, I have heard of a person who actually did this, a student who did this from IIT Delhi as a project. I am not sure whether he used machine learning or not but basically the idea was suppose you have an acceleration signal for your car and you suddenly have a rapid change, you can kind of figure out that this person has been in an accident or at least the cellphone can figure out that this person has been in an accident.

So a sequence to sequence classification for this would be at each point it will say moving, moving normally and at this point it will say, in an accident okay. So in such a case and in fact what this person did as I heard it was based on the acceleration signal of the cellphone, this person was able to alert the local ambulances or the local police authorities because apparently, the strongest saving of a life can be done at early times and by the time this person has an accident, somebody sees that this person is in an accident and calls the police or the ambulance or somebody, it is already sometimes too late.

But if the cellphone can figure this out and call immediately, then this is a great case. So I am going to show a variation of this, actually given by MATLAB, also based on accelerometer data within a person's pocket, okay. So let us see that case.

(Refer Slide Time 5:31)

Input
Horn
Horn

Online MATLAB

<https://in.mathworks.com/help/deeplearning/examples/sequence-to-sequence-classification-using-deep-learning.html>

LSTM - A type of RNN

NPTEL

So the example that I am going to show is from this link. This is both available within your online MATLAB account for the duration of this NPTEL course. So I am going to just introduce this case and I am not going to show you the training really, I am just going to show you snippets of the code from here and actually snippets of the description given by MATLAB itself. I would highly recommend that you actually go to your online MATLAB account and run this code. This is also available directly on MATLAB's website, the website that I have shown here.

Please take that, run that code and see for yourself how this works. There will be a future that will be used throughout this video, one of these terms will be LSTM. This is a term that I have not

explained so far. This is a type of RNN okay the most popular architecture of RNN available today. So the relationship between RNNs in LSTMs is somewhat similar to the relationship between CNNs and any particular architecture, let us say Lenet or Alex net, all those architectures that you saw.

So LSTM is a certain type of architecture of an RNN. So that is what is used within this example. So whenever you hear LSTM please think simply RNN okay.

(Refer Slide Time 6:56)

<https://in.mathworks.com/help/deeplearning/examples/sequence-to-sequence-classification-using-deep-learning.html>

LSTM → A type of RNN

Sequence-to-Sequence Classification Using Deep Learning

Time instant to activity prediction
5 classes

This example shows how to classify each time step of sequence data using a long short-term memory (LSTM) network.


To train a deep neural network to classify each time step of sequence data, you can use a sequence-to-sequence LSTM network. A sequence-to-sequence LSTM network enables you to make different predictions for each individual time step of the sequence data.

This example uses sensor data obtained from a smartphone worn on the body. The example trains an LSTM network to recognize the activity of the wearer given time series data representing accelerometer readings in three different directions. The training data contains time series data for seven people. Each sequence has three features and varies in length. The data set contains six training observations and one test observation.

a_x a_y a_z

Load Sequence Data

Load the human activity recognition data. The data contains seven time series of sensor data obtained from a smartphone worn on the body. Each sequence has three features and varies in length. The three features correspond to the accelerometer readings in three different directions.



Sequence-to-Sequence Classification Using Deep Learning

Time instant to activity prediction
5 classes

This example shows how to classify each time step of sequence data using a long short-term memory (LSTM) network.

To train a deep neural network to classify each time step of sequence data, you can use a sequence-to-sequence LSTM network. A sequence-to-sequence LSTM network enables you to make different predictions for each individual time step of the sequence data.

This example uses sensor data obtained from a smartphone worn on the body. The example trains an LSTM network to recognize the activity of the wearer given time series data representing accelerometer readings in three different directions. The training data contains time series data for seven people. Each sequence has three features and varies in length. The data set contains six training observations and one test observation.


Load Sequence Data

Load the human activity recognition data. The data contains seven time series of sensor data obtained from a smartphone worn on the body. Each sequence has three features and varies in length. The three features correspond to the accelerometer readings in three different directions.

Variable Length of Sequences
18000 length
9000 length
No. of features is the same

a_x a_y a_z

3600x5 = 18000



So let us come to this example. So this is from MATLAB's own description of this problem. So as it says that it is using LSTM which is, which stands for long short-term memory as we will see later and we will see the reason for this term also okay. So now what we are going to do is a sequel to sequence RNN or LSTM network. Basically what it takes is, it takes each time instant and it predicts what is the activity. So that is the basic task here. So time instant to activity prediction.

Now this can be tremendously useful, I told you one simple way of using it with let us say keeping an accelerometer within the car or keeping a smart phone within the car. But in this case we are taking a toy example, let us say this person has their cellphone attached to their body and based on the activity, the cellphone is supposed to figure out whether this person is sitting, standing, walking, dancing or running. Okay. So these are the 5 classes that we are going to look for.

So that is shown here. So the accelerometer is not measuring acceleration in only one direction. But it is actually measuring acceleration in 3 different directions. So the sensor gives you output in 3 different directions or accelerations in 3 different directions. And you measure this data for 7 different people, this is the training data. So as you can see, we are throwing up a lot of numbers. Let us sort of an disentangle them shortly, okay. okay.

Now where does the variable length come in? So each sequence has 3 features. What are these 3 features? Let us call them acceleration in direction X, acceleration in direction Y and acceleration in direction Z. So these are the 3 features that the sensor is throwing up at each time. But it varies in length. Why does it vary in length? As you will see shortly in a graph, what is it that we are measuring? What we are measuring is, suppose a person, let us say I wear this cellphone on my body or have it in my pocket, so let us say I draw a graph with time and I am doing some activity for a certain length of time. Okay.

So my acceleration signal looks like this with time in X, looks like this in time with Y and looks like this with time in Z. Now suppose I am wearing this accelerometer for let us say one hour. Okay. So the length of this sequence is suppose I am taking a signal every 5 seconds or so, okay. So if I am taking every 5 seconds or so, I will have 3600 times 5, I will have my length of my

sequence here because I will give one data point per 5 seconds, I am going to get 18,000 is the length of the sequence for the senses that I have worn.

Now let us see some other person, let us say you, you are wearing this but you might not wear it for one hour, you are wearing it for 30 minutes. In that case, your sequence length will be 9000. Somebody else wears it for something else, okay. So let us say that person wears it for 2 hours and that person's length is going to be 36,000. Now unlike CNNs where we were doing simple padding, it is sensible when if you get more data and you want a sequence to sequence prediction that you would like to take each one of these sequences and still give a prediction.

This is something that is not normally possible with ANNs or with CNNs. So variable length means your data, your time length can actually be different for different people. So let us see if you are trying to transcribe a youtube video and you are trying to say you know what has this person spoken or you have Alexa or you have Google, Amazon's Echo, something that is transcribing our speech, not all of us are going to speak for equal time instant. Each of us will have variable lengths.

Similarly you want to interpret or translate a sentence, each of these sentences is of different lengths. That is what is meant by variable length, okay. So this is an important property of an RNN that the length of the sequence can be different for different datasets. Okay. So each dataset can have different lengths. Even though the number of features is the same. okay. So please distinguish between the 2. When we see variable length, we mean the length of the time sequence over with this data is given.

That can be different for different samples, that is the advantage for RNNs, okay. And your final test data might be of an entirely different sequence length. So suppose I could say, I could just give 30 seconds of data and say what was this person doing? Okay. It would be a very short sequence thing but you could still be able to classify. Okay. So number of features however is always fixed. That is the case whether you have ANN, whether you have CNN or whether you have RNN, number of features we will fix. Okay. So this is the problem that we are starting with.

We are given AX with respect to T, AZ with respect to T, AY with respect to T and we want to predict at each time instant, what was this person doing. Such a task is called sequence to sequence classification task. Okay.

(Refer Slide Time 13:04)

Variable Length of sequence
9600 length
No. of features is the same

7 people
3 features $\rightarrow a_x, a_y, a_z$
Variable length
Equally spaced in time

```
load HumanActivityTrain
XTrain
```

So the way MATLAB does is it has its own dataset and I again, I recommend that you see this dataset from the site yourself and maybe exploded a little bit. So it has human activity training dataset. Remember, this has 7 people, 3 features-AX, AY, AZ and variable length. An important property of the RNN which I mentioned even in the last video is when we take different data points the structure of the RNN is such that these have to be equally spaced in time. Okay. For engineering problems, this I would recommend very strongly.

For engineering problems, it is best when you have, when you are genuinely dealing with time, as against let us say a language task where you know one word following the other is not really a time problem but when you have an actual engineering problem where time is one of the variables with respect to which you are doing an RNN, please try to use RNNs which are equally spaced in time or get datasets which are equally spaced in time. This is an important notion that you should be aware of. Okay.

So here we will assume that the sensor took its data at equal time instances okay. If you do not do this, it will be much harder to train it because of certain that features of the RNN which I will point out. Okay. So let us say we load this and we print what this X train is.

(Refer Slide Time 15:00)

The image shows a video lecture slide with a MATLAB terminal window and handwritten notes. The terminal window displays the command `load HumanActivityTrain XTrain` and the output `XTrain = 1x6 cell array` with six entries: `{3x64480 double}`, `{3x53696 double}`, `{3x56416 double}`, `{3x50688 double}`, `{3x51888 double}`, and `{3x54756 double}`. Handwritten notes in yellow and red ink include: "7 people - 1 testy", "3 features - a_x, a_y, a_z ", "Equally spaced in time", "Variable length", "6 people", "3 features for each data", and "Different sequence length". A small graph at the top shows a signal over time. An NPTEL logo is in the top right, and a "1/2" indicator is in the bottom right.

So you can see that this training set has 6 people. I had said 7 people but we are basically keeping one for testing. It is a small dataset. You cannot really do that 60-20-20 split, et cetera. So you take 6 people and tried to predict for one. Okay. So this training set is simply 6 people. Now notice, each of them has 3 features. So this is example 1, example 2, example 3 up till example 6. Okay. But each of them has different sequence length.

So person 1 actually measure or their activity or continued their activity for a much longer time because it is going for 64,480 whereas this person is probably the shortest in this dataset. This person went for 50,688, okay. So this is to say that you can train, even though each person actually gave you data for different lengths of time. Okay. So that is the advantage of an RNN. Okay.

(Refer Slide Time 16:25)



Let us now visualise the data. So we will just take person 1 and take one of the accelerometers or one of the directions or one of the sensors in the accelerometer and plot just that. Okay. So that is what is plotted here. You can see the time step, 10 to the power for up to approximately 6.4 into 10 to the power 4 , so $64,000$ datapoints. And we have the time signal, what we have plotted here is the signal of the accelerometer. Now as should be fairly obvious, when the person is sitting, you do not see very much acceleration.

Maybe this person like I am moving around, moved around a little bit okay. Now this dataset is labelled, what we have labelled with is you know we know the activity this person is doing. This person is sitting and at this point, at each point you say that the person was sitting. I will show you how to represent this in an RNN structure shortly. Okay. From this point to this point, the person got up and you can see, there is a rapid change in acceleration but as of now, we have still labelled it.

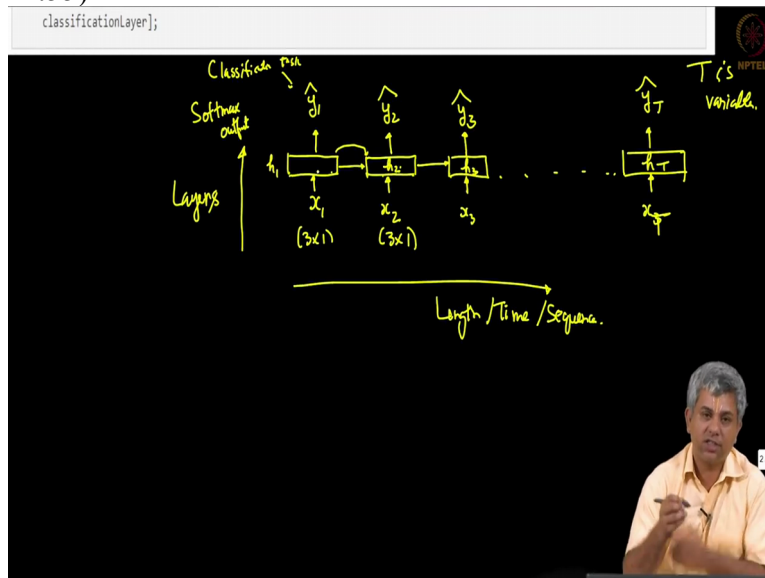
The label we gave was that of simply sitting. You will see the effect this has on the test set much later in a later video. Now at this point, this person is simply standing, that has the label here and you can see that even here the variation is not much. But also understand that somehow ed this thing is supposed to figure out you know how well this is doing based on here or here. Now when the person is walking, you can see a lot of variation in the person's acceleration, okay.

sequence to sequence classification task. I take the first time instead and somehow I am supposed to predict what \hat{Y}_1 is, okay.

Now the magic thing that is this layer in the middle, this is what we typically call H, H for hidden layer which is the terminology that we have used within neural networks right from the beginning. Now what is the size or what is the number of neurons in the hidden layer? I will call this H_1 , what is the number neurons in the hidden layer? This is once again, like we were doing before, it is our choice, okay. Only thing is, once we chose it, through the time sequence, we will keep this fixed. Okay.

So this is in this case, the MATLAB example has chosen 100 units but you can choose any number, obviously if you use smaller, you will have lesser expressibility but you will train faster okay. So we are going to choose 100 units for this particular example. So effectively, you can see this as 3 input neurons here, 100 hidden neurons and 5 output neurons. This is the neural network, except turned on its head like this okay. So we usually have neural network going this way, we have neural network going this way. Okay. Now what is an RNN?

(Refer Slide Time 21:55)



An RNN is multiple neural networks put together. Okay. So I start with X_1 , I have H_1 , get \hat{Y}_1 . But this H_1 is not wasted. It goes to another H_2 which also takes an input from X_2 . Okay so please notice what is happening here and see what in some sense makes RNNs work. What is it? Whatever output or whatever hidden layer, whatever features this hidden layer had, those are

not wasted, they are reused into the 2nd time instant, okay because you want to know what happened before. You do not want to forget that. We are not taking \hat{Y}_1 , remember all we are taking is the hidden layer input. Okay. This is sort of like what Dr Venapathy talked about in transfer learning.

I have already learned something, why waste that? I will transfer that information here. So this input goes here, H_1 goes here and H_2 is predicted, \hat{Y}_2 is predicted. This is like recursion. Not only do I want to know what happened now, I want to know what happened a little bit before. Then I take X_3 , \hat{Y}_3 . And the prediction method goes on like this. We will see later how to find out the losses and back propagation for such an architecture but you can go to some final time.

The point of an RNN is T is variable. When we see variable sequence, that is what we mean. So X_1 is 3 cross 1, X_2 is 3 cross 1, what does S_1 correspond to? X_1 corresponds to the 3 accelerometer signals at time 1, X_2 corresponds to the 3 accelerometer readings at time 2, so on and so forth, until time N or time T you collect the 3 accelerometer outputs okay. So what we had shown here was one accelerometer, similarly you will have 3 such things at each time instant. That is what you give here, okay.

Now what is \hat{Y}_1 , \hat{Y}_2 , \hat{Y}_3 ? You know this before, this is simply for a classification task. This is simply the soft max output. Okay. Another way to say it is, it is our approximation of a one hot vector, we know that this has to be either 01000 or 10000, et cetera. And we will approximate it through a soft max output. Now MATLAB in this particular example has chosen slightly differently. I will show you that. Before I show you, a little bit about terminology. Oftentimes in later videos, I will say number of layers. That is strictly speaking, not correct.

Number of layers should be in this direction. This is length, time or sequence. Okay. But I will sort of abuse notations and I will say you know when the number of layers increases, it is going to be very difficult to train but you should understand that number of layers actually in this direction. Right now, I have shown you a case with effectively one hidden layer. But just like we had neural networks before this which had more than one layer, you can simply do the same thing. You can take X , put let us say 2 layers and then get out Y . In fact, MATLAB has done something similar.

(Refer Slide Time 25:47)

Define LSTM Network Architecture

Define the LSTM network architecture. Specify the input to be sequences of size 3 (the feature dimension of the input data). Specify an LSTM layer with 100 hidden units, and output the full sequence. Finally, specify five classes by including a fully connected layer of size 5, followed by a softmax layer and a classification layer.

```
featureDimension = 3;
numHiddenUnits = 100;
numClasses = 5;

layers = [ ...
    sequenceInputLayer(featureDimension)
    lstmLayer(numHiddenUnits, 'OutputMode', 'sequence')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
```

Classification

If you see their description, after the hidden layer, there is a fully connected layer which you know what it means. So after this, so let us say I start with X_1 , I have a hidden layer with 100 neurons. After that I can put a fully connected layer of 5 neurons which means you know you are going to get a full fully connected thing here. After this, you have soft max. okay with 5. Okay. So you can make a soft Max layer after that, usually use our soft Max prediction formula and then go on from there.

But similarly, you can put 8 layers, 9 layers, typically people do not go more than 8 layers and these are for very complex tasks. I will show you a small example in a later video of a language task. Okay. But typically in layers, we go somewhere between for engineering tasks usually 1 or maximum 2 are good. Okay. So specially if you are dealing directly with numbers, 1 or 2 layers is usually good enough. And you will see that it actually takes a lot of effort to train such things. Okay.

(Refer Slide Time 27:08)

Diagram illustrating the RNN cell structure and matrix dimensions:

- Input: x_t (3x1)
- Previous hidden state: h_{t-1} (100x1)
- Hidden state: h_t (100x1)
- Output: \hat{y}_t (3x1)
- Length/Time/Sequence: t
- Equation: $h_t = \tanh \left(W_{xh} x_t + W_{hh} h_{t-1} + b \right)$
- Matrix dimensions:
 - W_{xh} : 100x3
 - W_{hh} : 100x100
 - x_t : 3x1
 - h_{t-1} : 100x1
 - b : 100x1

Handwritten notes and calculations:

- Calculation: $300 + 10,000 + 100 = 10,400$ parameters
- Equation: $W_{hh} h_{t-1} = 100 \times 100 = 10,000$
- Note: Same W_{xh} , W_{hh} , b for each time step → Power to the RNN

Now what is special about this RNN is remember that the weight that goes from length to length or time to time is exactly the same. Okay. Recollect if you have x_t , h_{t-1} , h_t coming out and let us say \hat{y}_t , then h_t is let us say I will say \tanh in this but some nonlinearity of some weight matrix multiplying x_t + some other weight matrix multiplying h_{t-1} . The subscripts are as follows. For x_t I will call it W_{xh} . For h_{t-1} I will call it W_{hh} and of course, you have to add a bias term.

Now for this example, let us see what the sizes of these matrices ought to be. Okay. So let us see this. x_t at any point is a 3 cross 1 matrix, h is a 100 cross 1 matrix. So what should be the size

of WXH ? Please think about it. Similarly, $HT - 1$ is a 100 cross 1 matrix, HT is also a 100 cross 1 matrix, what should be the size of WHH ? Please think about it. HT is a size of 100 cross 1, what should be B ? okay.

If you are thinking about it, please pause the video for a second, think about it, work it out and then come back, I will show you the answer after you are done. Okay. I hope you tried it. So as you can see, WXH takes in a 3 cross 1 matrix, okay. So this is 3 cross 1, this should give you a 100 cross 1 matrix as output which means WXH has to be 100 cross 3. Similarly, WHH has to be 100 cross 100 so that it can take a 100 cross 1 matrix and give a 100 cross 1 matrix as output. Simple dimensional consistency or length consistency requires that you have 100 biases for a 100 cross 1 matrix here, okay.

Now giving all this is fine. How many parameters do we have? You have 300 parameters here, 10,000 parameters here and another 100 parameters here. Okay. So you have $300 + 10,000 + 100$. So which is 10,400 parameters for this problem. Now you may question that you have 10,400 parameters but then what happens at the next time step? Okay. The catch here and which is why I said that in engineering examples, it is a good idea to take the same ΔT between this and this, this and this is that we will assume that it is the same WXH and WHH and B for each time, okay.

So the only W used W s you use are the W s that you will use for the first layer, they remain a constant as you go across and RNN and that is the power of an RNN. Otherwise, the number of parameters would explode. So from time to time to time to time, you are going to assume that the matrices do not change, that is the relationship between H_3 and H_2 is the same as the relationship between H_4 and H_3 is the same as the relationship between H_5 and H_4 . This basically is what gives power to the RNN.

More than power, this is what gives compactness. Okay. So once you use this, you have your RNN, you have just set your structure this way. There are various ways of compactly representing RNNs. We are not going to deal with that in this course, this is just to give you an introduction. I would recommend very strongly that even before you see the final video, you run this particular case here and see if you are able to understand what is going on. You can in fact

see here that there is an LSTM layer which is made up of a fully connected layer and a soft max layer and finally the classification layer which will tell you how it goes.

(Refer Slide Time 32:39)

Diagram of an LSTM cell showing inputs h_{t-1} and x_t , and output h_t .

Calculations for the number of parameters:

$$W_{xh} x_t = 100 \times 3 \times 1 = 100 \times 1$$
$$W_{hh} h_{t-1} = 100 \times 100 \times 1 = 100 \times 1$$

Total parameters: $300 + 10,000 + 100 = 10,400$ parameters

Backprop + Loss

Same W_{xh}, W_{hh}, b for each time \rightarrow Power RNN

Hopefully you also saw the number parameters that pop up in this problem, in this case 10,400 parameters and you have to train all these 10,400 parameters via back prop. Now how do you do back prop for this case? How do you calculate loss in such a case? This we will see in the next few videos, thank you.