



Follow

606K Followers

·

Editors' Picks

Features

Deep Dives

You have **2** free member-only stories left this month. [Sign up for Medium and get an extra one](#)

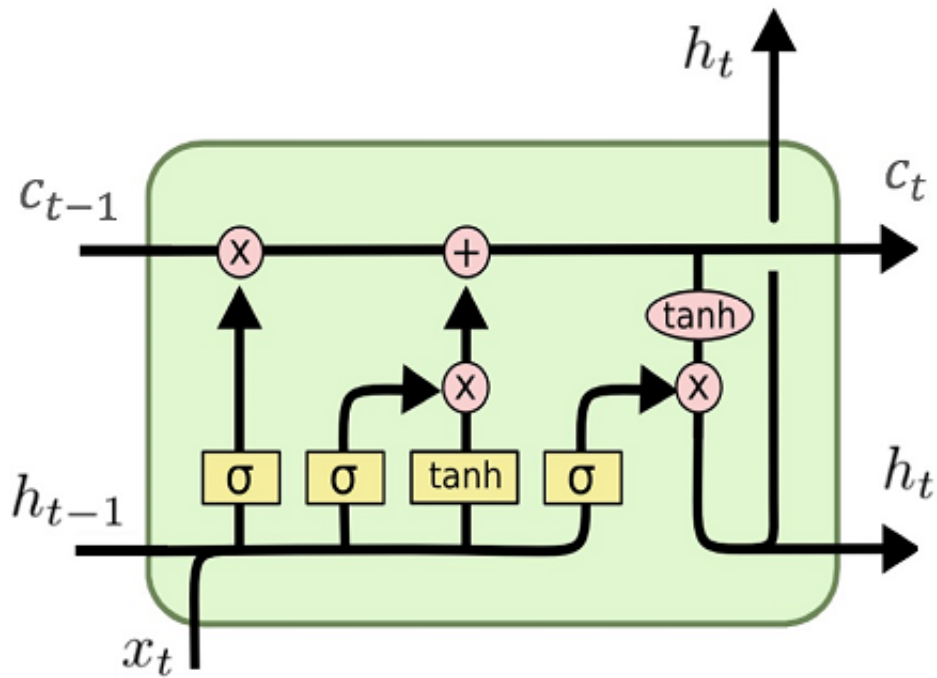
LSTM Gradients

Detailed mathematical derivation of gradients for LSTM cells.



Rahuljha Jun 29, 2020 · 11 min read ★

LSTM or Long Short Term Memory is a very important building block of complex and state of the art neural network architectures. The main idea behind this article is explaining the math behind it. To get an initial understanding of what LSTM is, I would suggest the following blog.



LSTM (Long-Short Term Memory)

[Credits](#)

Understanding LSTM Networks

Posted on August 27, 2015 Humans don't start their thinking from scratch every second. As you read this essay, you...

colah.github.io

Contents :

A — Concept

- Introduction
- Explanation

- Derivation Prerequisites

B — Derivation

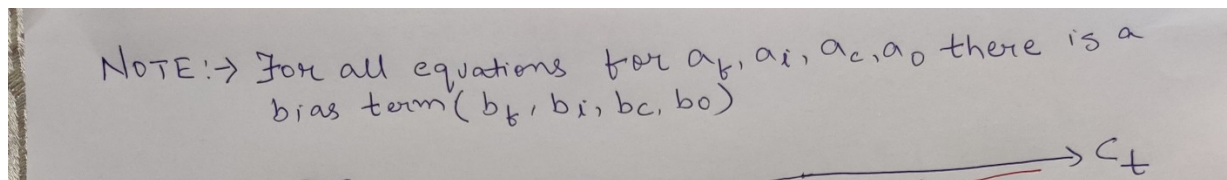
- Output of the LSTM
- Hidden state
- Output gate
- Cell state
- Input gate
- Forget gate
- Input to the LSTM
- Weights and biases

C — Back propagation through time

D — Conclusion

Concept

Introduction



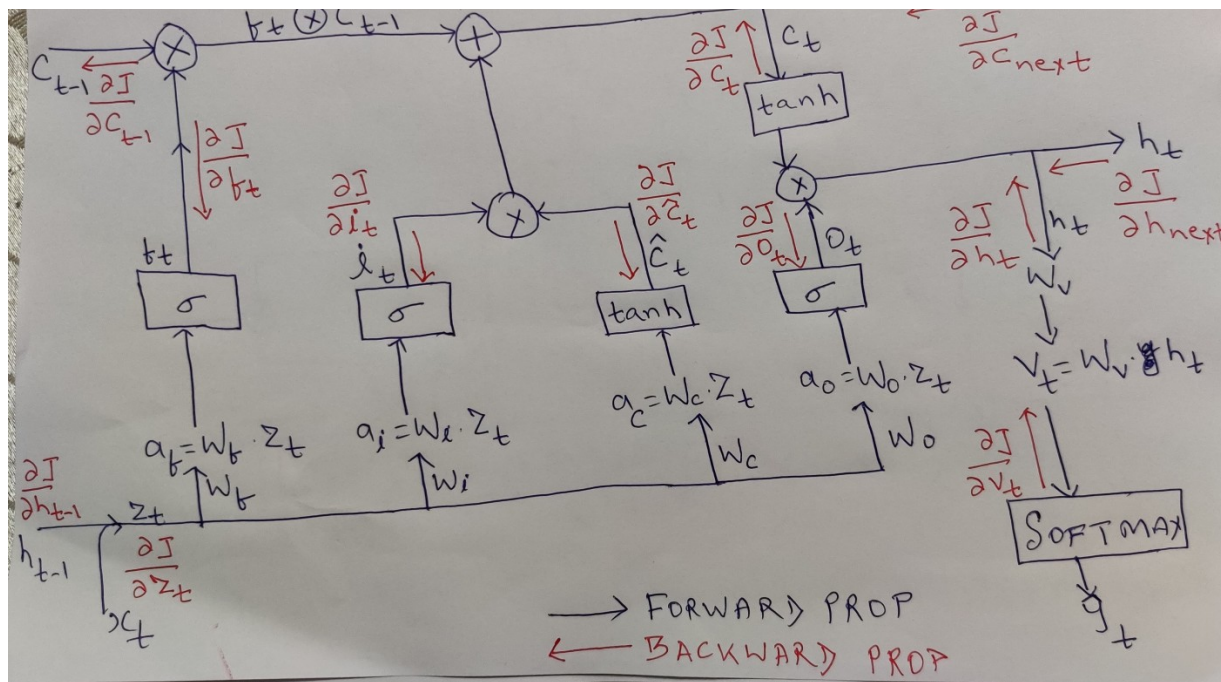


Fig 1: LSTM Cell

The above is a diagram for a single LSTM cell. I know it looks scary 😬, but we will go through it one by one and by the end of the article, hopefully it will be pretty clear.

Explanation

Basically a single LSTM cell has 4 different components. Forget gate, input gate, output gate and the cell state. We will first discuss the use of these parts in brief (for detailed explanation please refer to the above blog) and then dive into the math part of it.

Forget gate

As the name suggests, this part is responsible for deciding what information is to be thrown away or kept from the last step. This is done by the first sigmoid layer.

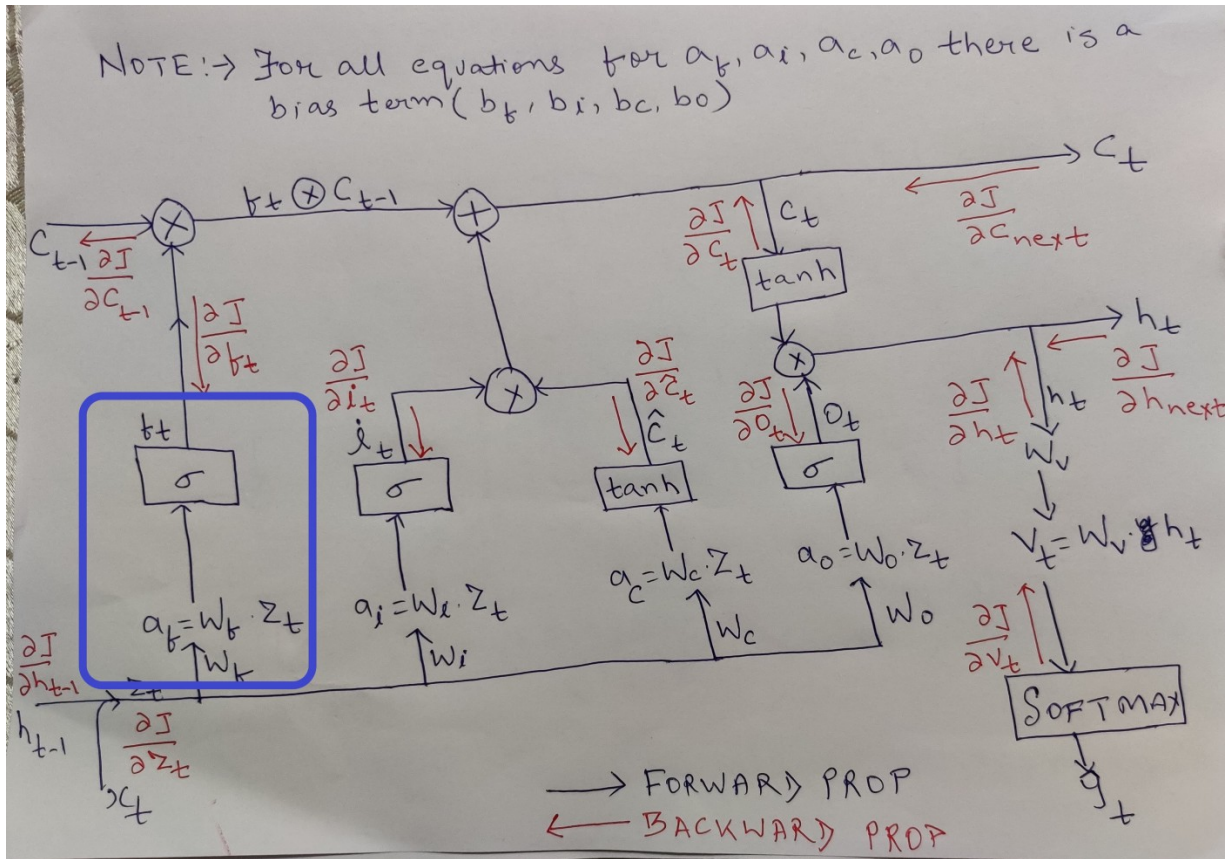


Fig 2: Forget gate marked in Blue

Based on h_{t-1} (previous hidden state) and x_t (current input at time-step t), this decides a value between 0 and 1 for each value in cell state C_{t-1} .



Fig 3: Forget gate and previous cell state

For all 1's, all the information is kept as it is, for all 0's all the information is discarded and with other values it decides how

much information from previous state is to be carried to the next state.

Input gate

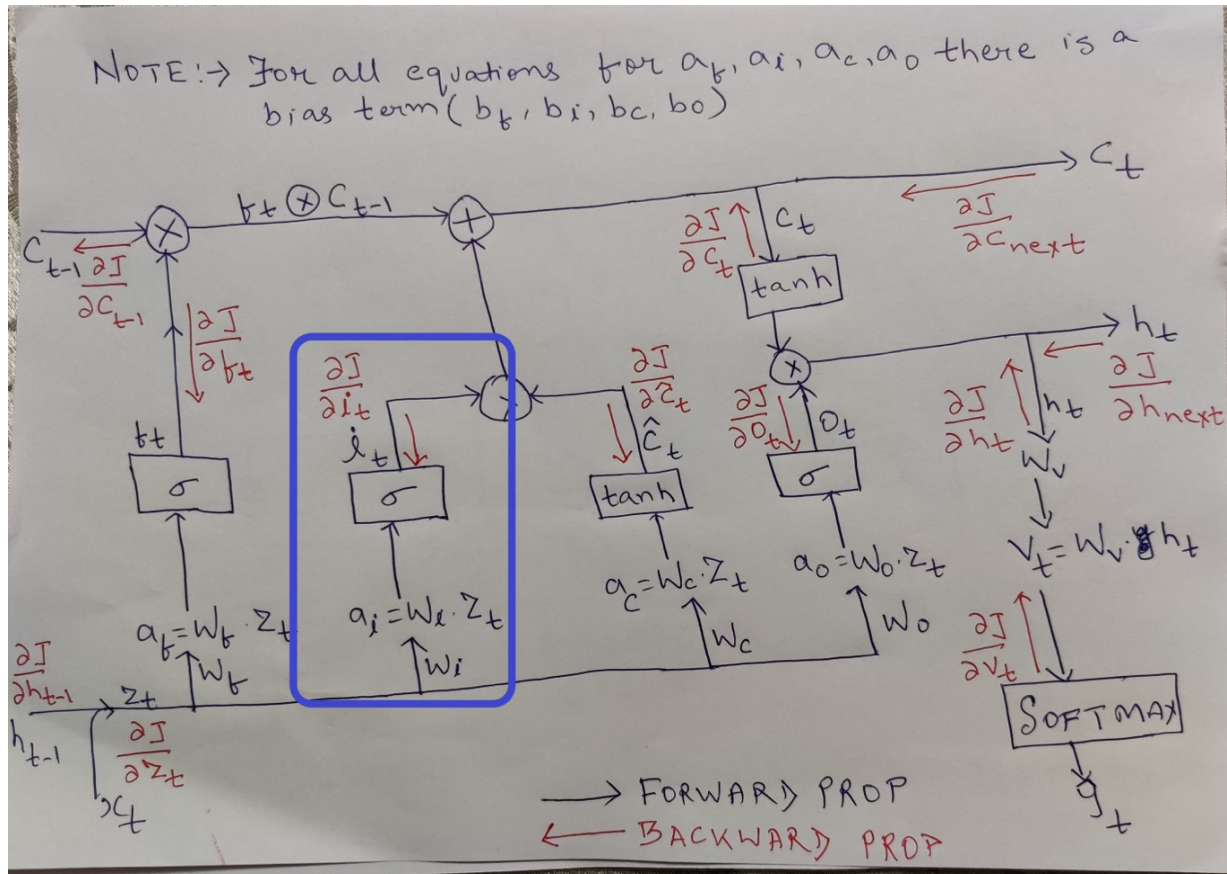


Fig 4: Input gate marked in Blue

Christopher Olah has a beautiful explanation of what happens in the input gate. To cite his blog:

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $C_{\sim t}$, that could be added to the state. In the next step, we'll combine

these two to create an update to the state.

Now these two values i.e i_t and c_t combine to decide what new input is to be fed to the cell state.

Cell state

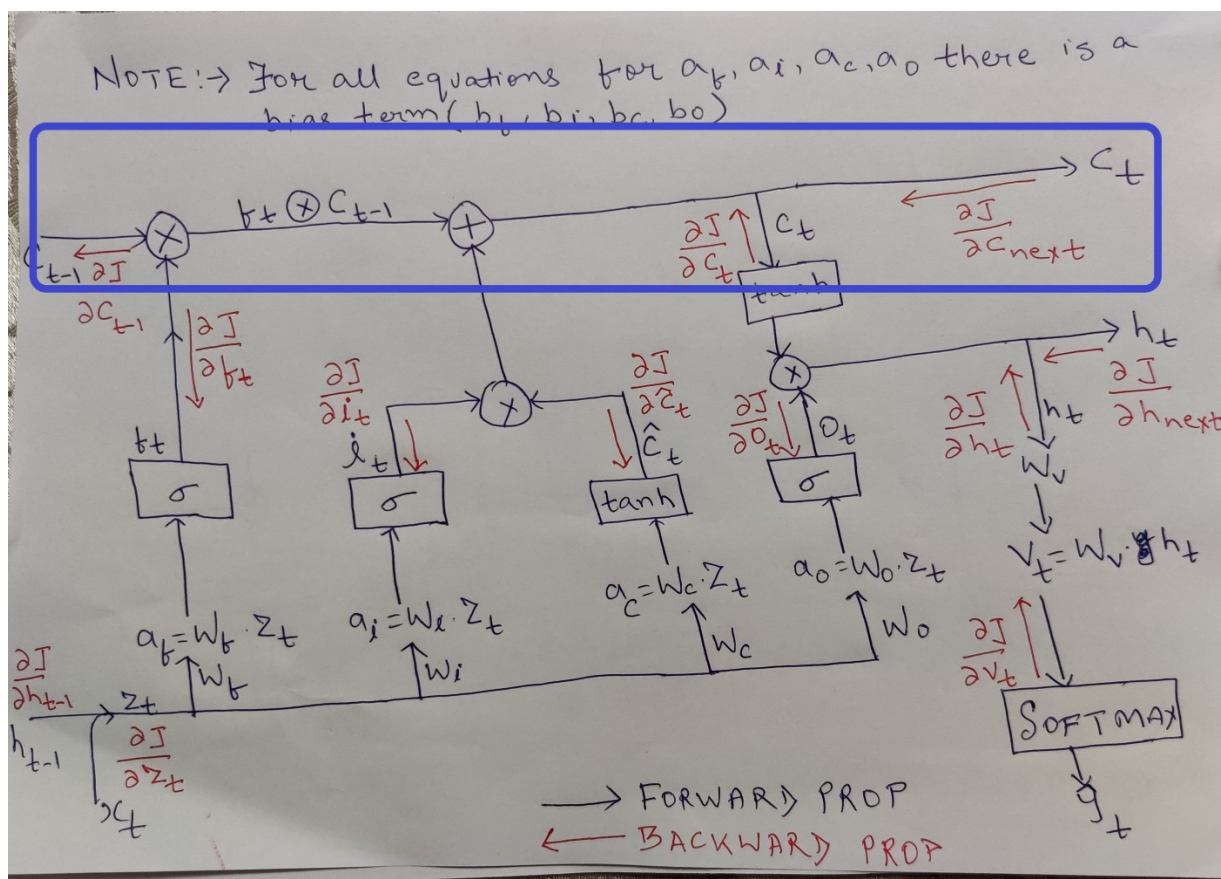


Fig 5: Cell state marked in Blue

Cell state serves as the memory of an LSTM. This is where they perform way better than vanilla RNN's when dealing with longer sequences of input. At each time-step the previous cell state (c_{t-1}) combines with the forget gate to decide what information is to be carried forward which in turn combines with the input gate (i_t and c_t) to form the new cell state or

the new memory of the cell.



Fig 6: New cell state equation

Output gate

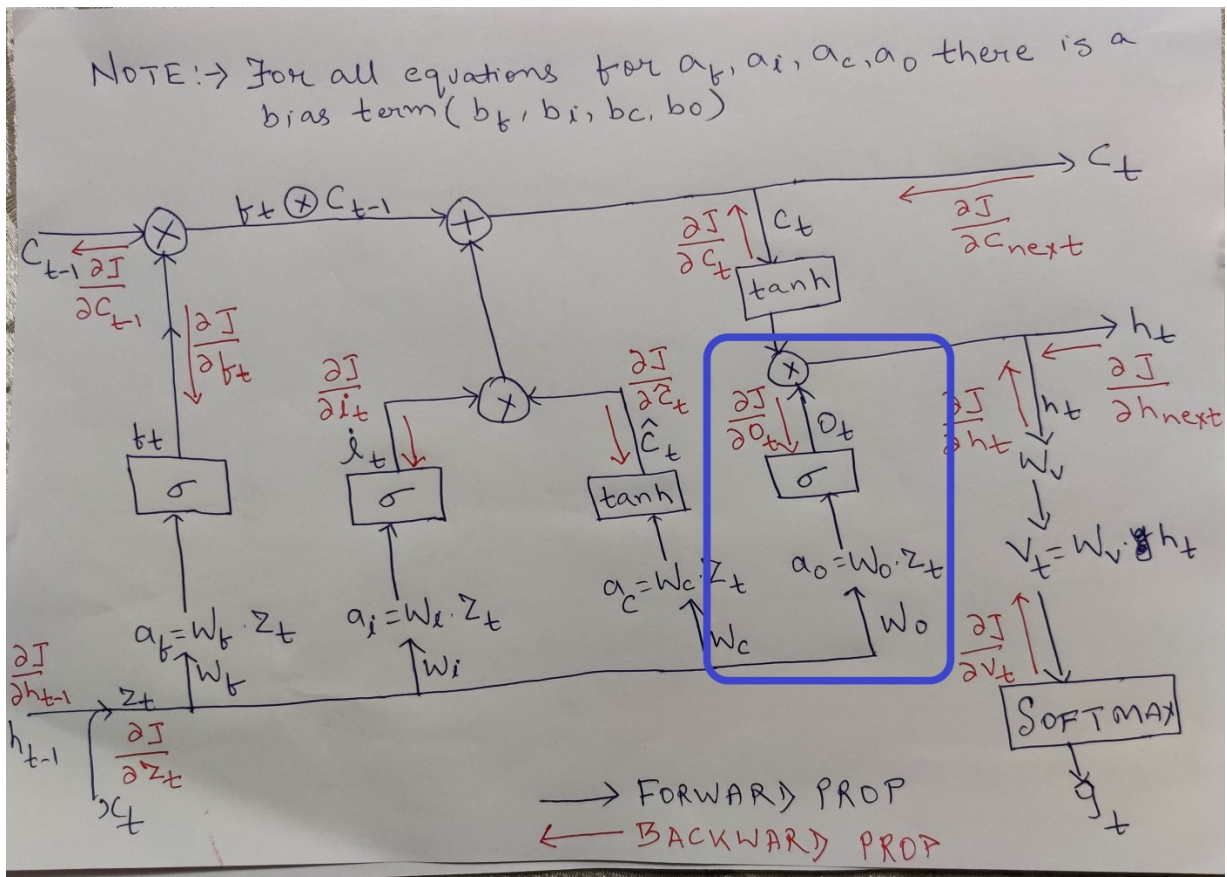


Fig 7: Output gate marked in Blue

At last the LSTM cell has to give some output. The cell state obtained from above is passed through a hyperbolic function called tanh so that the cell state values are filtered between -1

and 1. For details into different activation function, [this](#) is a nice blog.

Now i hope the basic cell structure of a LSTM cell is clear and we can proceed to the derivation of equations which we will use in our implementation.

Derivation Prerequisites

1. **Requirements** : The core concept of deriving equations is based on backpropagation, cost function and loss. If you are not familiar with these , these are few links that will help in getting a good understanding. This article also assumes a basic understanding of high school calculus (calculating derivatives and there rules).

Understanding Backpropagation Algorithm

Learn the nuts and bolts of a neural network's most important ingredient

towardsdatascience.com

Finding the Cost Function of Neural Networks

Part 1 of Step by Step: The Math Behind Neural Networks

towardsdatascience.com

2. **Variables** : For each gates we have a set of weights and

biases which will be denoted as:

- W_f, b_f -> Forget gate weight and bias
- W_i, b_i -> Input gate weight and bias
- W_c, b_c -> Candidate cell state weight and bias
- W_o, b_o -> Output gate weight and bias

W_v, b_v -> Weight and bias associated with the Softmax layer.

f_t, i_t, c_t, o_t -> Output of the activation functions

a_f, a_i, a_c, a_o -> Input to the activation functions

J is the cost function, with respect to which we will be calculating the derivatives. Note the (character after the underscore(_) is a subscript)

3. Forward prop equations:

$$\text{Forget gate} : a_f = W_f \cdot Z_t + b_f \quad f_t = \text{sigmoid}(a_f) \quad 1$$

$$\text{Input gate} : a_i = W_i \cdot Z_t + b_i \quad i_t = \text{sigmoid}(a_i) \quad 2$$

$$: a_c = W_c \cdot Z_t + b_c \quad \tilde{c}_t = \text{tanh}(a_c) \quad 3$$

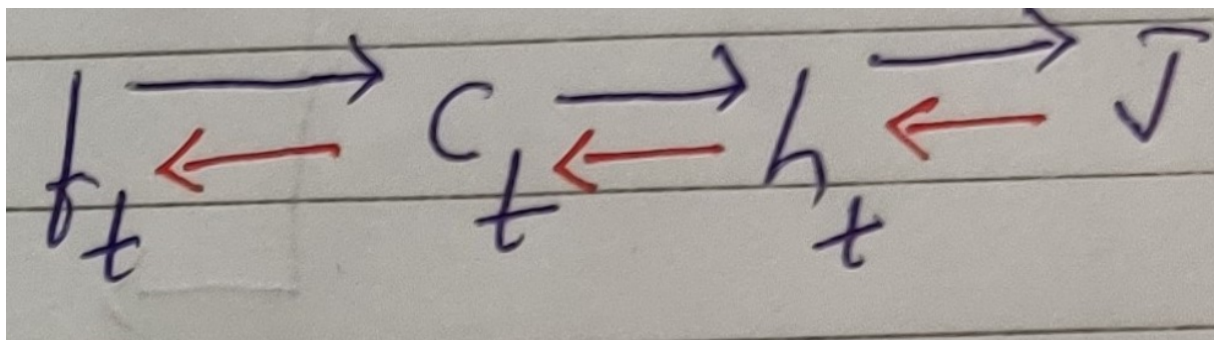
$$\text{Output gate} : a_o = W_o \cdot Z_t + b_o \quad o_t = \text{sigmoid}(a_o) \quad 4$$

Fig 8: Gate equations

$$\begin{aligned}
 \text{Cell state} & : C_t = f_t \otimes C_{t-1} \oplus i_t \otimes \tilde{c}_t & 5 \\
 \text{Hidden state} & : h_t = o_t \otimes \tanh(c_t) & 6 \\
 \text{Output equations} & : V_t = W_v \cdot h_t + b_t & 7 \\
 & \hat{y}_t = \text{softmax}(V_t) & 8
 \end{aligned}$$

Fig 9: Cell state and output equations

4. **Process for calculation** : Let's take forget gate example to illustrate the calculation of the derivatives. We need to follow the path of red arrows in the below figure.



So we chalk out a path from f_t to our cost function J i.e

$$f_t \rightarrow C_t \rightarrow h_t \rightarrow J.$$

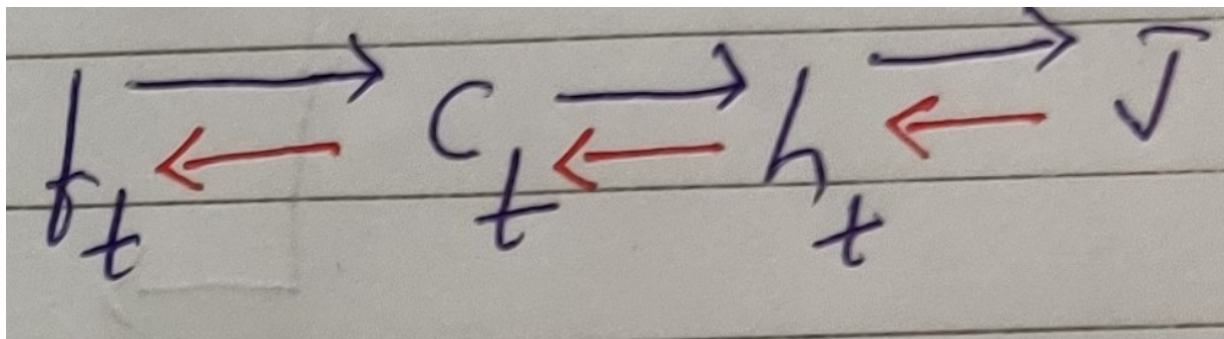
The backpropagation happens exactly in the same step but in reverse i.e

$$f_t \leftarrow C_t \leftarrow h_t \leftarrow J.$$

J is differentiated with respect to h_t , h_t with respect to C_t and C_t with respect to f_t .

So if we observe here, J and h_t is the last step of the cell, and if we calculate dJ/dh_t , then it can be used for calculations like dJ/dC_t since :

$$dJ/dC_t = dJ/dh_t * dh_t/dC_t \text{ (Chain rule)}$$



Similarly, the derivatives will be calculated for all the variables mentioned in point no 1.

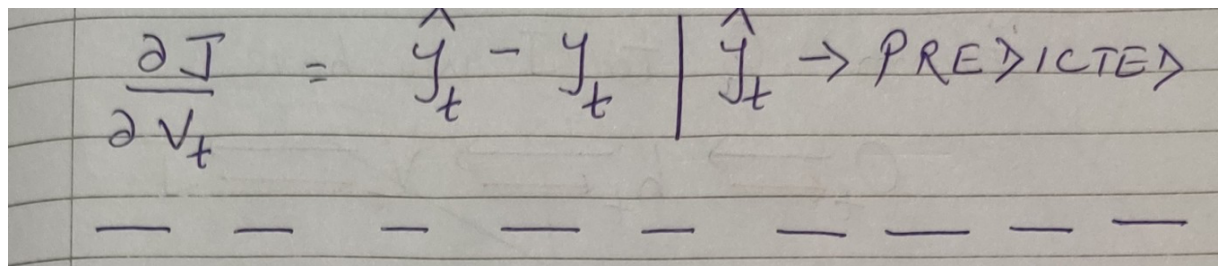
Now that we have the variables ready and we are clear with the forward prop equations, its time to dive into deriving the derivatives through back-propagation. We will start with the output equations as we saw that the same derivatives is used in other equations. This is where the chain rule comes in. So let's start now.

Derivation

Output of the lstm

The output has two values which we need to calculate.

1. Softmax : For derivative of Cross Entropy Loss with Softmax we will be using the final equation directly.


$$\frac{\partial J}{\partial v_t} = \hat{y}_t - y_t \quad | \quad \hat{y}_t \rightarrow \text{PREDICTED}$$

The detailed derivation can be found below:

A Gentle Introduction to Cross-Entropy Loss Function - Sefik Ilkin Serengil

Neural networks produce multiple outputs in multi-class classification problems. However, they do not...

sefiks.com

Hidden State

We have the hidden state as h_t . h_t is differentiated w.r.t J . According to chain rule, the derivation can be seen in the below figure. We use the value of V_t as mentioned in Fig 9 equation 7 i.e :

$$V_t = W_v \cdot h_t + b_v$$

Hidden State

$$\frac{\partial J}{\partial h_t} = \frac{\partial J}{\partial v_t} \cdot \frac{\partial v_t}{\partial h_t} \quad (\text{Chain Rule})$$

$$= \frac{\partial J}{\partial v_t} \cdot \frac{\partial}{\partial h_t} (w_v \cdot h_t + b_v)$$

$$= \frac{\partial J}{\partial v_t} \cdot \frac{\partial}{\partial h_t} (w_v \cdot h_t) \Rightarrow w_v^T \cdot \frac{\partial J}{\partial v_t}$$

Output gate

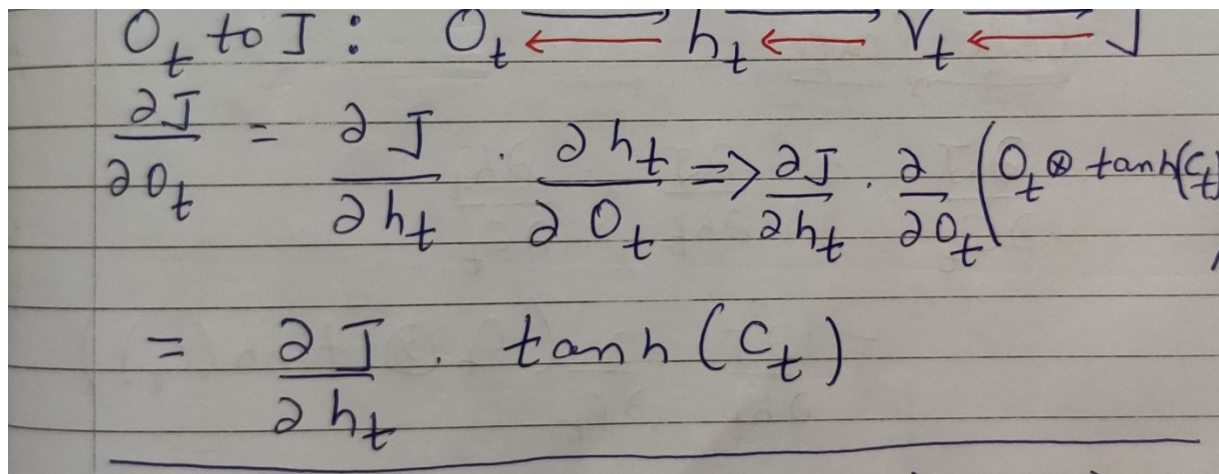
Variables associated : a_o and o_t .

o_t : In the below image, the path between o_t and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dv_t * dv_t/dh_t * dh_t/dO_t$$

$dJ/dv_t * dv_t/dh_t$ can be written as dJ/dh_t (we have this value from hidden state).

The value of $h_t = o_t * \tanh(c_t) \rightarrow$ Fig 9 equation 6. **So we only need to differentiate h_t w.r.t o_t .** The differentiation will be as :-



$$O_t \text{ to } J: O_t \xrightarrow{h_t} V_t \xrightarrow{J}$$

$$\frac{\partial J}{\partial O_t} = \frac{\partial J}{\partial h_t} \cdot \frac{\partial h_t}{\partial O_t} \Rightarrow \frac{\partial J}{\partial h_t} \cdot \frac{\partial (O_t \otimes \tanh(c_t))}{\partial O_t}$$

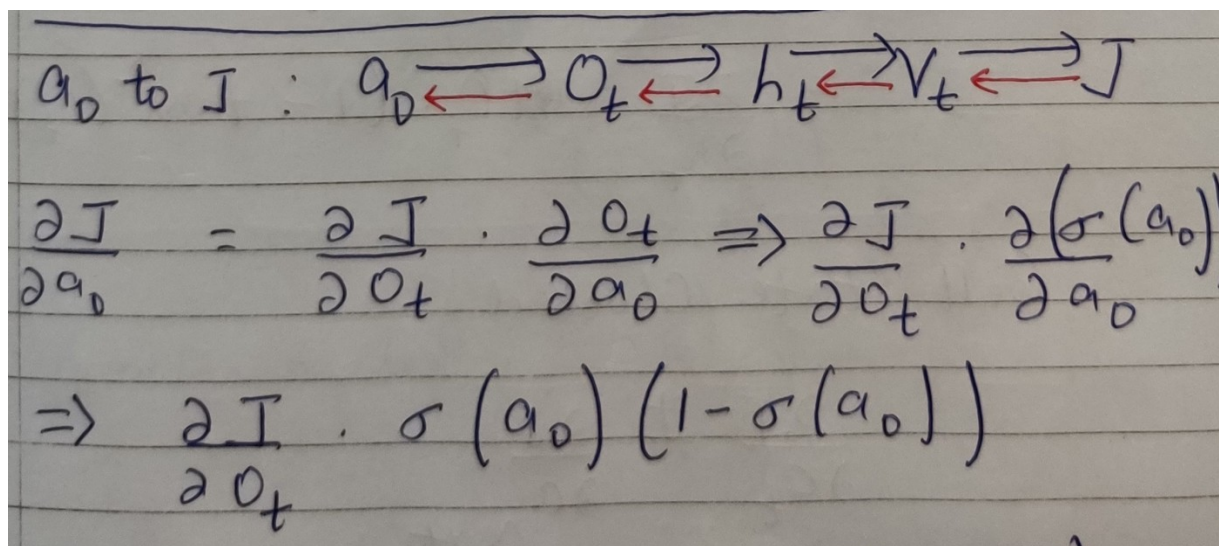
$$= \frac{\partial J}{\partial h_t} \cdot \tanh(c_t)$$

a_o: Similarly, the path between a_o and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dV_t * dV_t/dh_t * dh_t/dO_t * dO_t/da_o$$

$dJ/dV_t * dV_t/dh_t * dh_t/dO_t$ can be written as dJ/dO_t (we have this value from above o_t).

$o_t = \text{sigmoid}(a_o) \rightarrow$ Fig 8 equation 4. **So we only need to differentiate o_t w.r.t a_o .** The differentiation will be as :-



$$a_o \text{ to } J: a_o \xrightarrow{O_t} h_t \xrightarrow{V_t} J$$

$$\frac{\partial J}{\partial a_o} = \frac{\partial J}{\partial O_t} \cdot \frac{\partial O_t}{\partial a_o} \Rightarrow \frac{\partial J}{\partial O_t} \cdot \frac{\partial (\sigma(a_o))}{\partial a_o}$$

$$\Rightarrow \frac{\partial J}{\partial O_t} \cdot \sigma(a_o) (1 - \sigma(a_o))$$

$$\Rightarrow \frac{\partial J}{\partial h_t} \cdot \tanh(c_t) \cdot o_t(1 - o_t)$$

Cell State

C_t is the cell state of the cell. Along with it, we also handle the candidate cell state a_c and $c_{\sim t}$ here.

C_t : The derivation for C_t is pretty trivial, as the path from C_t to J is simple enough. $C_t \rightarrow h_t \rightarrow V_t \rightarrow J$. As we already have dJ/dh_t , we directly differentiate h_t w.r.t C_t .

$h_t = o_t * \tanh(c_t) \rightarrow$ Fig 9 equation 6. **So we only need to differentiate h_t w.r.t C_t .**

$$\begin{aligned} \frac{\partial J}{\partial C_t} &= \frac{\partial J}{\partial h_t} \cdot \frac{\partial h_t}{\partial C_t} \\ &= \frac{\partial J}{\partial h_t} \cdot \frac{\partial (o_t \otimes \tanh(c_t))}{\partial C_t} \\ &= \frac{\partial J}{\partial h_t} \cdot o_t \cdot (1 - \tanh^2(c_t)) \end{aligned}$$

Cell state Clubbed:-

$$\frac{\partial J}{\partial C_t} \pm \frac{\partial J}{\partial C_{next}}$$

Note: The cell state clubbed will be explained at the end of the article.

$c_{\sim t}$: In the below image, the path between $c_{\sim t}$ and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dh_t * dh_t/dC_t * dC_t/dc_{\sim t}$$

$dJ/dh_t * dh_t/dC_t$ can be written as dJ/dC_t (we have this value from above).

The value of C_t is as shown in Fig 9 equation 5 (tilde (\sim) sign is missing in the last c_t in line no 3 in below figure -> writing mistake). **So we only need to differentiate C_t w.r.t $c_{\sim t}$.**

$$\hat{c}_t \text{ to } J \rightarrow \hat{c}_t \longleftrightarrow C_t \longleftrightarrow h_t \longleftrightarrow J$$

$$\frac{\partial J}{\partial \hat{c}_t} = \frac{\partial J}{\partial C_t} \cdot \frac{\partial \hat{c}_t}{\partial \hat{c}_t}$$

$$\frac{\partial C_t}{\partial \hat{c}_t} = \frac{\partial}{\partial \hat{c}_t} (f_t \otimes C_{t-1} \oplus i_t \otimes c_t)$$

$$= i_t$$

$$\frac{\partial J}{\partial \hat{c}_t} = \frac{\partial J}{\partial C_t} \cdot i_t$$

a_c : In the below image, the path between a_c and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dh_t * dh_t/dC_t * dC_t/dc_{\sim t} * dc_{\sim t}/da_c$$

$dJ/dh_t * dh_t/dC_t * dC_t/dc_{\sim t}$ can be written as $dJ/dc_{\sim t}$ (we have this value from above).

The value of $c_{\sim t}$ is as shown in Fig 8 equation 3. **So we only need to differentiate $c_{\sim t}$ w.r.t a_c .**

Diagram showing the path of gradients: $a_c \rightarrow \hat{c}_t \rightarrow C_t \rightarrow h \rightarrow J$. Red arrows indicate the backward flow of gradients.

$$\frac{\partial J}{\partial a_c} = \frac{\partial J}{\partial \hat{c}_t} \cdot \frac{\partial \hat{c}_t}{\partial a_c} = \frac{\partial J}{\partial \hat{c}_t} \cdot \frac{\partial (\tanh(a_c))}{\partial a_c}$$

$$= \frac{\partial J}{\partial \hat{c}_t} \cdot (1 - \tanh(a_c)^2)$$

$$= \frac{\partial J}{\partial C_t} \cdot i_t \cdot (1 - \hat{c}_t^2) \quad \boxed{\frac{\partial J}{\partial \hat{c}_t} = \frac{\partial J}{\partial C_t} \cdot i_t}$$

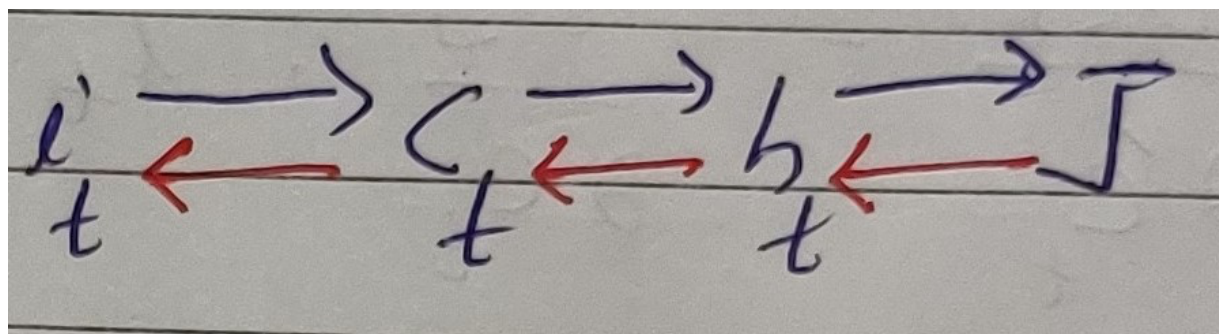
Input gate

Variables associated : i_t and a_i

i_t : In the below image, the path between i_t and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dh_t * dh_t/dC_t * dC_t/di_t$$

$dJ/dh_t * dh_t/dC_t$ can be written as dJ/dC_t (we have this value from cell state). **So we only need to differentiate C_t w.r.t i_t .**



The value of C_t is as shown in Fig 9 equation 5. So the differentiation will be as :-

$$\begin{aligned} \frac{\partial J}{\partial i_t} &= \frac{\partial J}{\partial C_t} \cdot \frac{\partial C_t}{\partial i_t} \\ &= \frac{\partial J}{\partial C_t} \cdot \frac{\partial}{\partial i_t} (b_t \otimes C_{t-1} \oplus i_t \otimes \hat{C}_t) \\ &= \frac{\partial J}{\partial C_t} \cdot \hat{C}_t \end{aligned}$$

a_i : In the below image, the path between a_i and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dh_t * dh_t/dC_t * dC_t/di_t * di_t/da_i$$

$dJ/dh_t * dh_t/dC_t * dC_t/di_t$ can be written as dJ/di_t (we have this value from above). **So we only need to differentiate i_t w.r.t a_i.**

$a_i \rightarrow J \rightarrow a_i \rightarrow i_t \rightarrow C_t \rightarrow h_t \rightarrow J$

$$\frac{\partial J}{\partial a_i} = \frac{\partial J}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i} = \frac{\partial J}{\partial C_t} \cdot \hat{C}_t \cdot \frac{\partial (\sigma(a_i))}{\partial a_i}$$

$$\frac{\partial (\sigma(a_i))}{\partial a_i} = \sigma(a_i) (1 - \sigma(a_i))$$

$$\because \sigma(a_i) = i_t$$

$$\frac{\partial J}{\partial a_i} = \frac{\partial J}{\partial C_t} \cdot \hat{C}_t \cdot i_t (1 - i_t)$$

Forget Gate

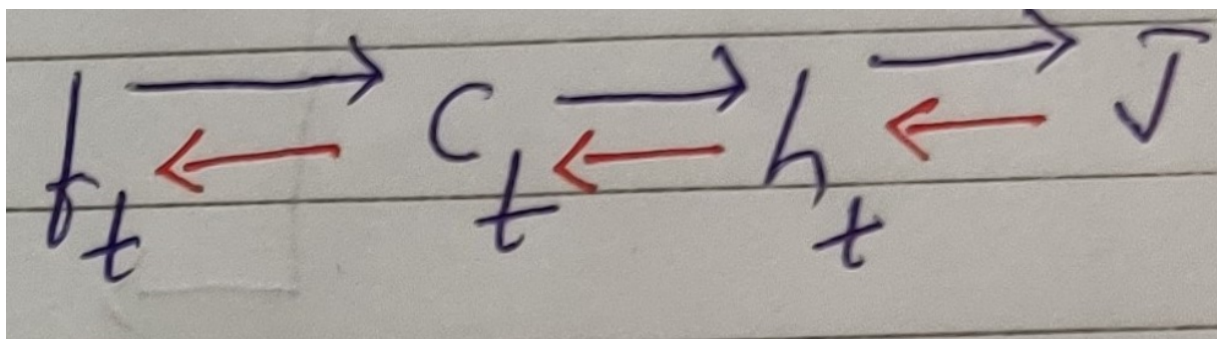
Variables associated : f_t and a_f

f_t : In the below image, the path between f_t and J is shown.

According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dh_t * dh_t/dC_t * dC_t/df_t$$

$dJ/dh_t * dh_t/dC_t$ can be written as dJ/dC_t (we have this value from cell state). **So we only need to differentiate C_t w.r.t f_t .**



The value of C_t is as shown in Fig 9 equation 5. So the differentiation will be as :-

$$\begin{aligned} \frac{\partial J}{\partial f_t} &= \frac{\partial J}{\partial C_t} \cdot \frac{\partial C_t}{\partial f_t} \\ &= \frac{\partial J}{\partial C_t} \cdot \frac{\partial}{\partial f_t} (b_t \otimes c_{t-1} \oplus i_t \otimes \hat{c}_t) \\ &= \frac{\partial J}{\partial C_t} \cdot c_{t-1} \end{aligned}$$

a_f : In the below image, the path between f_t and J is shown. According to the arrows the full equation for the differentiation will be as follows:

$$dJ/dh_t * dh_t/dC_t * dC_t/df_t * df_t/da_t$$

$dJ/dh_t * dh_t/dC_t * dC_t/df_t$ can be written as dJ/df_t (we have this value from above). **So we only need to differentiate f_t w.r.t a_f .**

$$\begin{aligned}
 \frac{\partial J}{\partial a_f} &= \frac{\partial J}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f} \\
 &= \frac{\partial J}{\partial C_t} \cdot C_{t-1} \cdot \frac{\partial (\sigma(a_f))}{\partial a_f} \\
 &= \frac{\partial J}{\partial C_t} \cdot C_{t-1} \cdot \sigma(a_f) (1 - \sigma(a_f)) \\
 &= \frac{\partial J}{\partial C_t} \cdot C_{t-1} \cdot b_t (1 - f_t)
 \end{aligned}$$

Input to the Lstm

There are 2 variables associated with input for each cell i.e previous cell state C_{t-1} and previous hidden state concatenated with current input i.e

$[h_{t-1}, x_t] \rightarrow Z_t$

C_{t-1} : This is the memory of the Lstm cell. Figure 5 shows the cell state. The derivation of C_{t-1} is pretty simple as only C_{t-1} and C_t are involved.

$$\frac{\partial J}{\partial C_{t-1}} = \frac{\partial J}{\partial C_t} \cdot \frac{\partial C_t}{\partial C_{t-1}} \Rightarrow \frac{\partial J}{\partial C_t} \cdot b_t$$

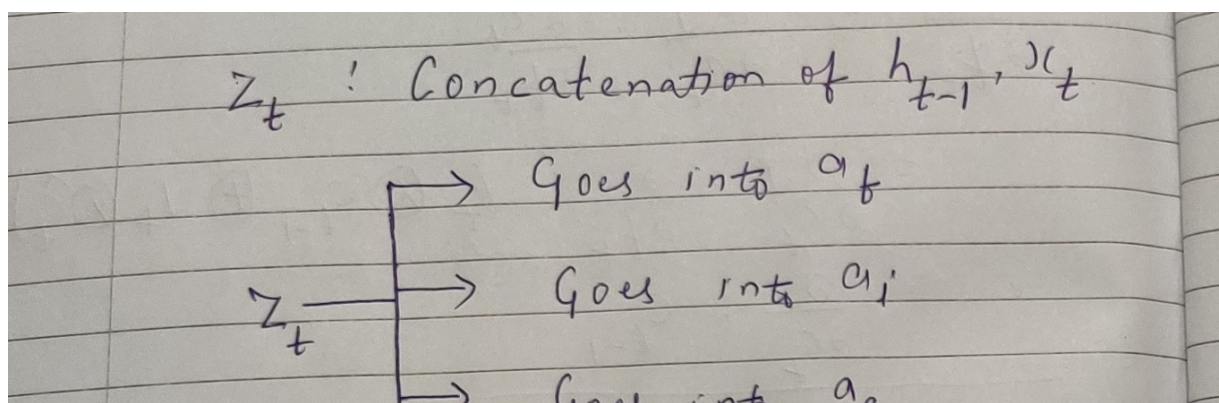
Z_t : As shown in the below figure, Z_t goes into 4 different path, a_f, a_i, a_o, a_c .

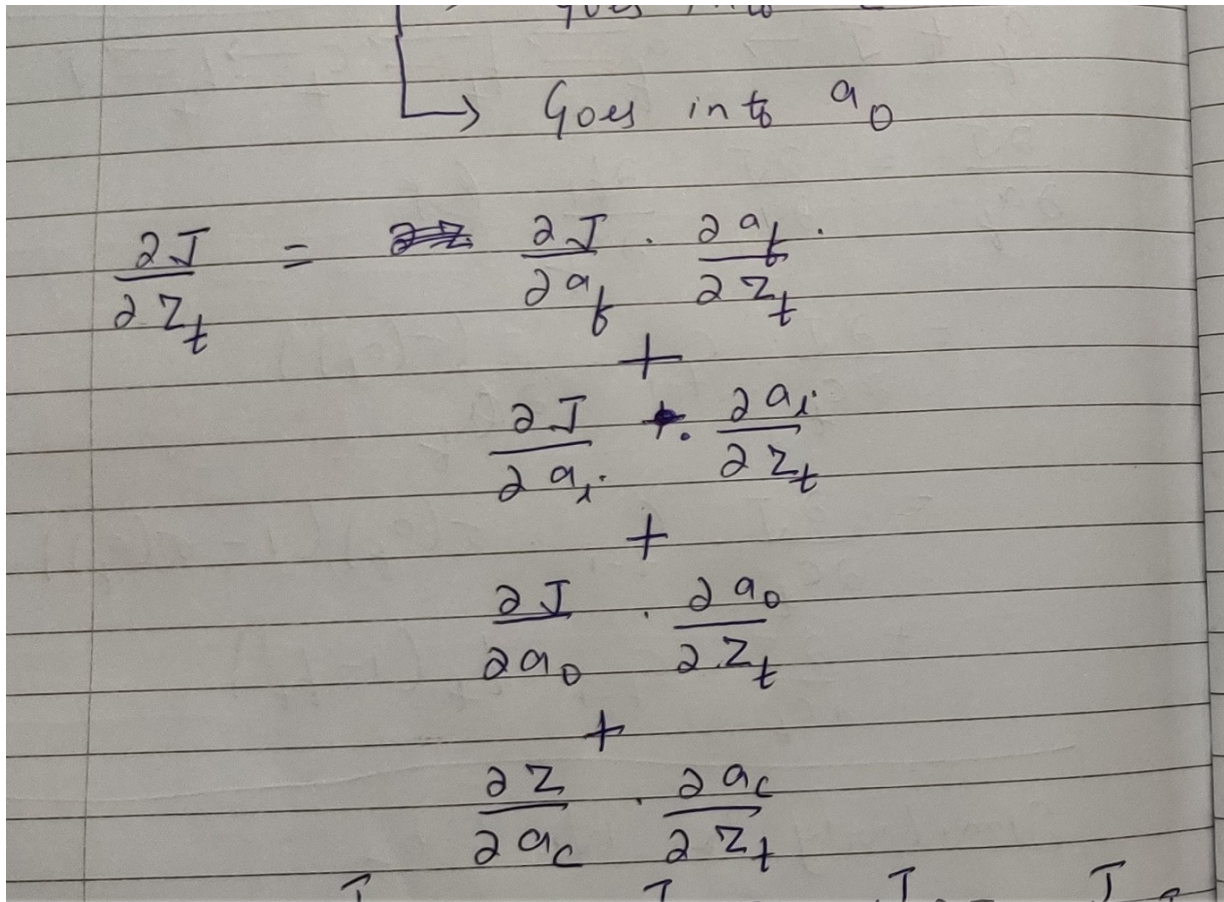
$Z_t \rightarrow a_f \rightarrow f_t \rightarrow C_t \rightarrow h_t \rightarrow J$. -> Forget gate

$Z_t \rightarrow a_i \rightarrow i_t \rightarrow C_t \rightarrow h_t \rightarrow J$. -> Input gate

$Z_t \rightarrow a_c \rightarrow c_{\sim t} \rightarrow C_t \rightarrow h_t \rightarrow J$. -> Candidate cell state

$Z_t \rightarrow a_o \rightarrow o_t \rightarrow C_t \rightarrow h_t \rightarrow J$. -> Output gate





To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

Rahuljha

Follow biases

The derivation for W and b is straight forward. The below derivation is for the output gate of the Lstm. For the rest of the gates, similar process is done for weights and biases.

RAHULJHA, FOLLOWS

Isha Thakur

Shipra Saxena

Karolis Ramanauskas

Himanshu Rajput

Dennis Ieros

See all (8)

95 4

Similarly for W_f & b_f

$$\frac{\partial J}{\partial W_f} = \frac{\partial J}{\partial a_f} \cdot \frac{\partial a_f}{\partial W_f} \Rightarrow \frac{\partial J}{\partial a_f} \cdot \sum_t^T$$

$$\frac{\partial J}{\partial b_f} = \frac{\partial J}{\partial a_f} \cdot \frac{\partial a_f}{\partial b_f} \Rightarrow \frac{\partial J}{\partial a_f}$$

Similarly for W_i & b_i

$$\frac{\partial J}{\partial W_i} = \frac{\partial J}{\partial a_i} \cdot \frac{\partial a_i}{\partial W_i} \Rightarrow \frac{\partial J}{\partial a_i} \cdot \sum_t^T$$

$$\frac{\partial J}{\partial b_i} = \frac{\partial J}{\partial a_i} \cdot \frac{\partial a_i}{\partial b_i} \Rightarrow \frac{\partial J}{\partial a_i}$$

Weights and biases for input and forget gate

$$\frac{\partial J}{\partial W_v} = \frac{\partial J}{\partial v_t} \cdot \frac{\partial v_t}{\partial W_v} \text{ (chain rule)}$$

$$\frac{\partial J}{\partial W_v} = \frac{\partial J}{\partial v_t} \cdot \frac{\partial}{\partial W_v} (W_v \cdot h_t + b_v)$$

$$= h_t^T \cdot \frac{\partial J}{\partial v_t}$$

Similarly for b_v :

$$\frac{\partial J}{\partial b_v} = \frac{\partial J}{\partial v_t} \cdot \frac{\partial v_t}{\partial b_v}$$

$$= \frac{\partial J}{\partial v_t} \cdot \frac{\partial}{\partial b_v} (W_v \cdot h_t + b_v)$$

$$= \frac{\partial J}{\partial v_t}$$

$$\frac{\partial J}{\partial W_o} = \frac{\partial J}{\partial a_o} \cdot \frac{\partial a_o}{\partial W_o}$$

$$= \frac{\partial J}{\partial a_o} \cdot \frac{\partial}{\partial W_o} (W_o \cdot z_t + b_o)$$

$$= \frac{\partial J}{\partial a_o} \cdot \sum_t^T$$

Similarly $\frac{\partial J}{\partial b_o} = \frac{\partial J}{\partial a_o} \cdot \frac{\partial a_o}{\partial b_o} = \frac{\partial J}{\partial a_o}$

Weights and bias for output and output gate

$dJ/d_{W_f} = dJ/da_f \cdot da_f/d_{W_f} \rightarrow$ Forget gate

$dJ/d_{W_i} = dJ/da_i \cdot da_i/d_{W_i} \rightarrow$ Input gate

$dJ/d_{W_v} = dJ/dv_t \cdot dv_t/d_{W_v} \rightarrow$ Output

$dJ/d_{W_o} = dJ/da_o \cdot da_o/d_{W_o} \rightarrow$ Output gate

So finally we are done with all the derivations. Now we just need to clarify a couple of points.

Back propagation through time

Till now what we have done is for a single time step. Now we have to make it for 1 single iteration.

So if we have total T time-steps, then the gradients for each single time-step will be added at the end of T time steps, so the cumulative gradient at end of each iteration will be:

$$\begin{aligned}
 dJ / dW_f &= \text{summation } t \text{ to } T (dJ / dW^{t_f}) \\
 dJ / dW_i &= \text{summation } t \text{ to } T (dJ / dW^{t_i}) \\
 dJ / dW_o &= \text{summation } t \text{ to } T (dJ / dW^{t_o}) \\
 dJ / dW_c &= \text{summation } t \text{ to } T (dJ / dW^{t_c}) \\
 dJ / dW_v &= \text{summation } t \text{ to } T (dJ / dW^{t_v})
 \end{aligned}$$

Fig 10 : Cumulative gradient at end of each iteration

And now these will be used to update the Weights.

$$W_f += \text{alpha} * dJ/dW_f$$

$$W_i += \text{alpha} * dJ/dW_i$$

$$W_c += \text{alpha} * dJ/dW_c$$

$$W_o += \alpha * dJ/dW_o$$
$$W_v += \alpha * dJ/dW_v$$

alpha = Learning rate

Fig 11: Weight updation

Conclusion

LSTMs are very complex structures but they also work very well. Mainly there are 2 types of RNN's with this features: LSTM and GRU.

Training of LSTMs is also a tricky task as there are lots of hyper-parameters and getting the combination right is often a difficult task.

So, by now i hope the math part of LSTM is pretty clear and i would suggest getting your hands dirty to get a clear and good understanding of it. Here are some links which might help:

Understanding LSTM and its quick implementation in keras for sentiment analysis.

towardsdatascience.com

Keras LSTM tutorial - How to easily build a powerful deep learning language model -...

In previous posts, I introduced Keras for building convolutional neural networks and performing wor...

adventuresinmachinelearning.com

Thanks for reading the article and hope you liked it.

Happy learning 😊 !!!

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Get this newsletter

Lstm

Gradient

Backpropagation

Deep Learning

Derivation

[About](#) [Write](#) [Help](#) [Legal](#)