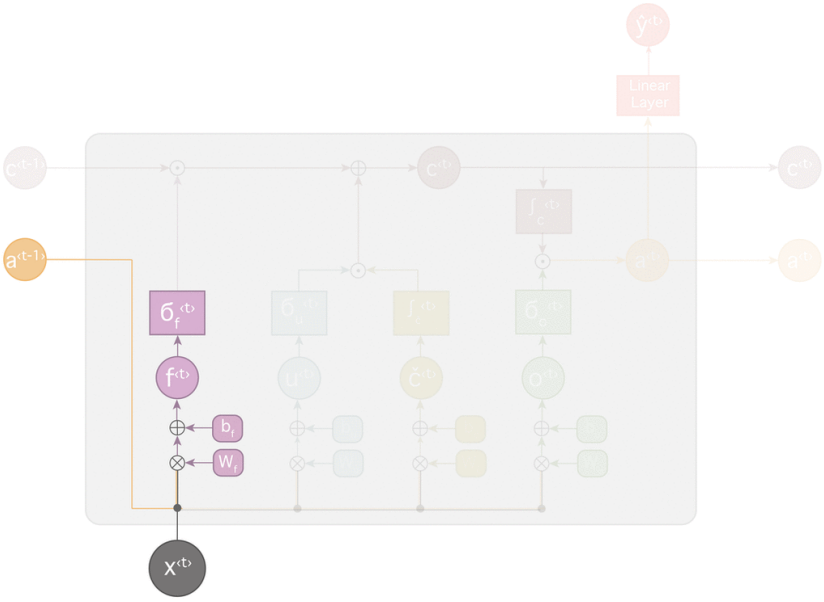
Eniola Alese Follow
I dont know so I learn and write.
Jun 8 · 4 min read

# And of course, LSTM — Part I

## The ABC's of the LSTM

In our last post we looked at the reasons why gradients vanish and presented the LSTM architecture as one of the solutions for dealing with the problem. In this post, we would look at the LSTM's internal structure and gain an intuitive understanding of its computations.




LSTM Unit

The LSTM, just like in a standard RNN receives its input from the current time-step input `x<t>` and from the previous time-step hidden state activation `a<t-1>` . The main structural differences between the two units are:

- the introduction of a memory cell state `c<t>` ,
- introduction of three sigmoid gates (forget gate `σf<t>,` update gate `σu<t>,` output gate `σo<t>` ),

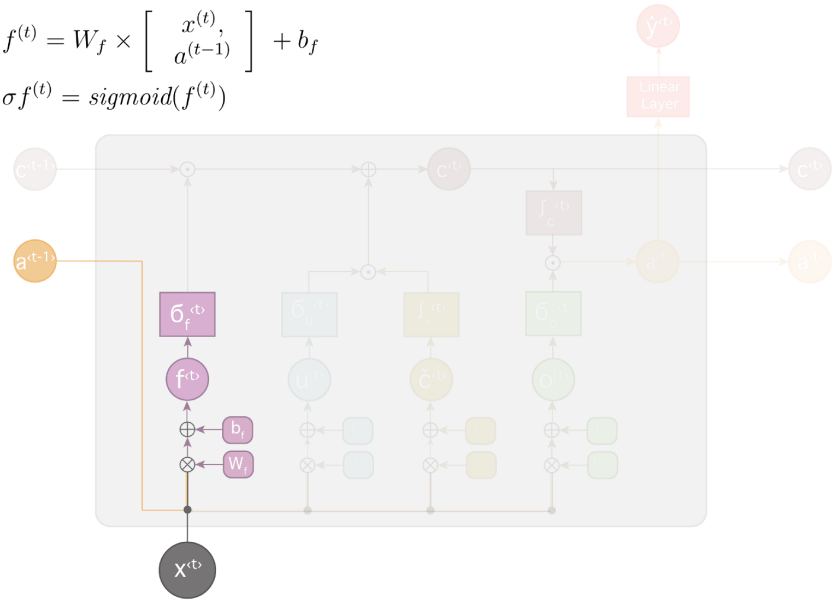
- and the ability to remove or add information to the memory cell state.

Lets break down what happens into individual steps:

**Step 1:** The forget gate `σf<t>` acts like a switch and decides whether or not to discard the memory cell state information that came from the previous time-step `c<t-1>`. It does this by applying a linear transformation to its inputs `x<t>` and `a<t-1>`, and then passes the output through a logistic function, which then outputs a value between 0 and 1.

$$f^{(t)} = W_f \times \begin{bmatrix} x^{(t)}, \\ a^{(t-1)} \end{bmatrix} + b_f$$

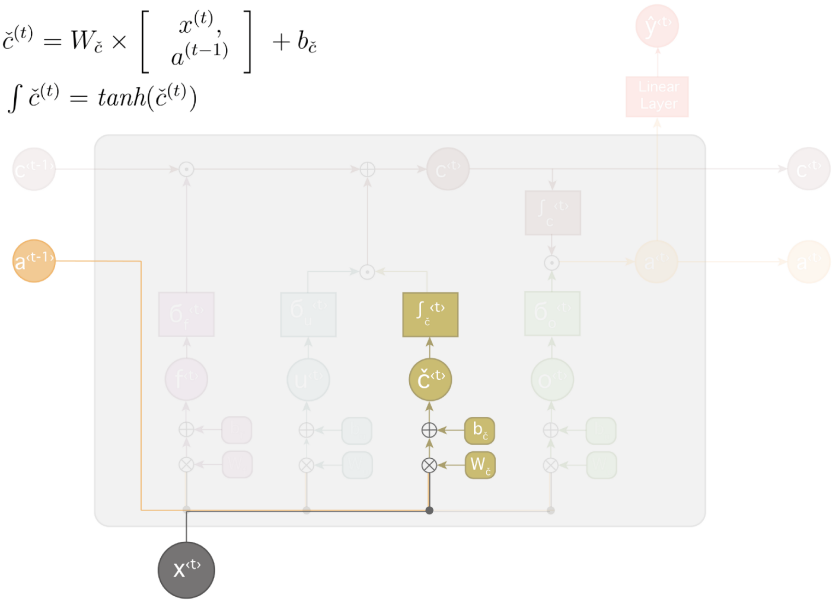
$$\sigma f^{(t)} = sigmoid(f^{(t)})$$




**Step 2:** Next, calculate the temporary memory activation `ʃč<t>` which holds the temporary memory cell information about the current time-step. This is computed by passing the inputs `x<t>` and `a<t-1>` through a linear layer and then through a tanh activation function which then outputs a value between -1 and 1.

$$\check{c}^{(t)} = W_{\check{c}} \times \begin{bmatrix} x^{(t)}, \\ a^{(t-1)} \end{bmatrix} + b_{\check{c}}$$

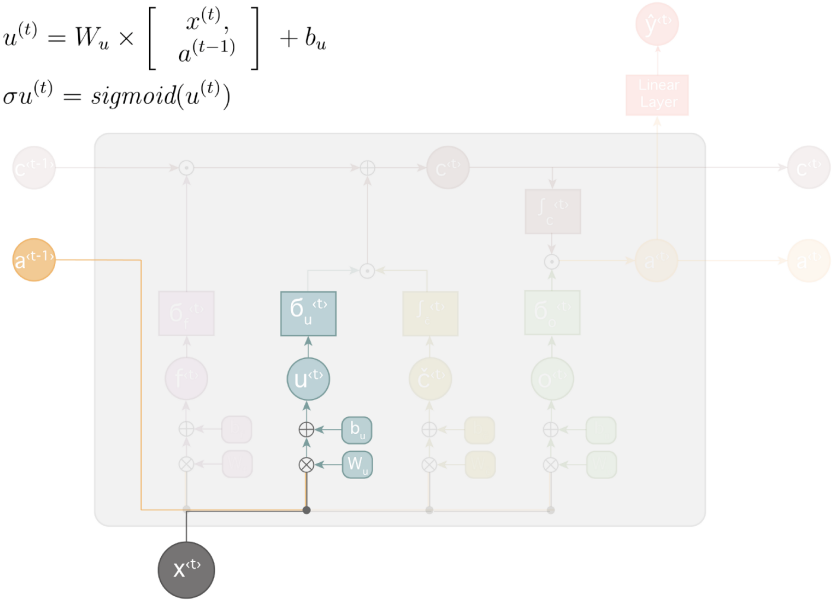
$$\int \check{c}^{(t)} = tanh(\check{c}^{(t)})$$



**Step 3:** Next we have the update gate `σu<t>` which has the same computational form as the forget gate (i.e linear-to-logistic) and like the forget gate its job is to decide whether or not to discard the temporary memory activation `∫č<t>` .

$$u^{(t)} = W_u \times \begin{bmatrix} x^{(t)}, \\ a^{(t-1)} \end{bmatrix} + b_u$$

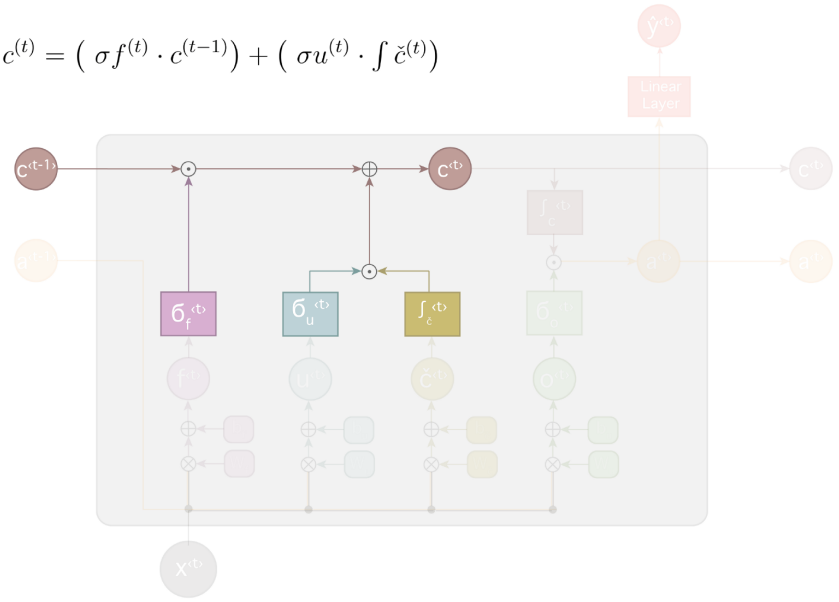
$$\sigma u^{(t)} = sigmoid(u^{(t)})$$



**Step 4:** Next we update the memory cell state `c<t>` which is the core of the LSTM design because its what enables the LSTM remember information over long distance. We do this by multiplying the cell state information that came from the previous time-step `c<t-1>` by the

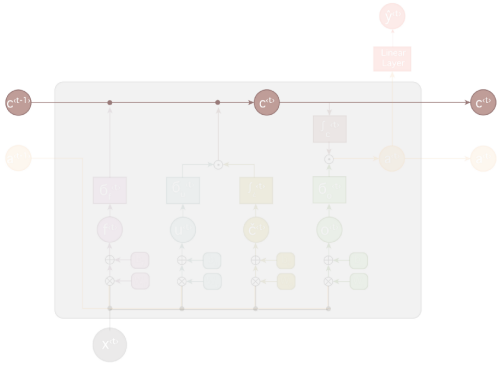
forget gate `σf<t>` and adding this to the multiplication of the update gate `σu<t>` and the temporary memory activation `∫č<t>` .

$$c^{(t)} = \left( \sigma f^{(t)} \cdot c^{(t-1)} \right) + \left( \sigma u^{(t)} \cdot \int \check{c}^{(t)} \right)$$



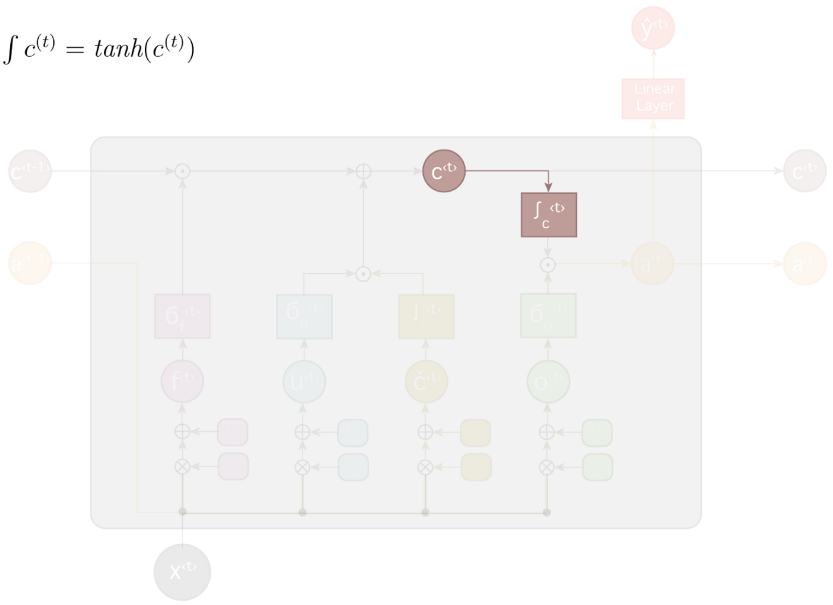
Depending on the values of the forget and update gates the memory cell state can exhibit various behaviours, like if the forget gate `σf<t>` is ON (1) and the update gate `σu<t>` is OFF (0) the memory cell state would compute something close to the identity function: `c<t> = c<t-1>`

| $\sigma f^{(t)}$ | $\sigma u^{(t)}$ | $c^{(t)}$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | $c^{(t-1)}$ |
| 0 | 1 | $\int \check{c}^{(t)}$ |
| 1 | 1 | $c^{(t-1)} + \int \check{c}^{(t)}$ |



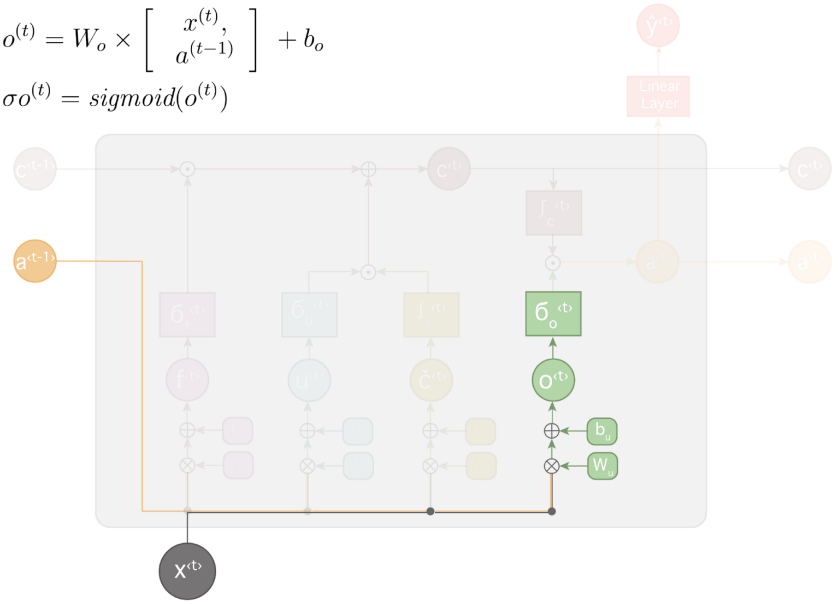
**Step 5**: Next, we compute the memory activation `∫c<t>` which is simply the memory cell state `c<t>` passed through a tanh function.

$$\int c^{(t)} = tanh(c^{(t)})$$



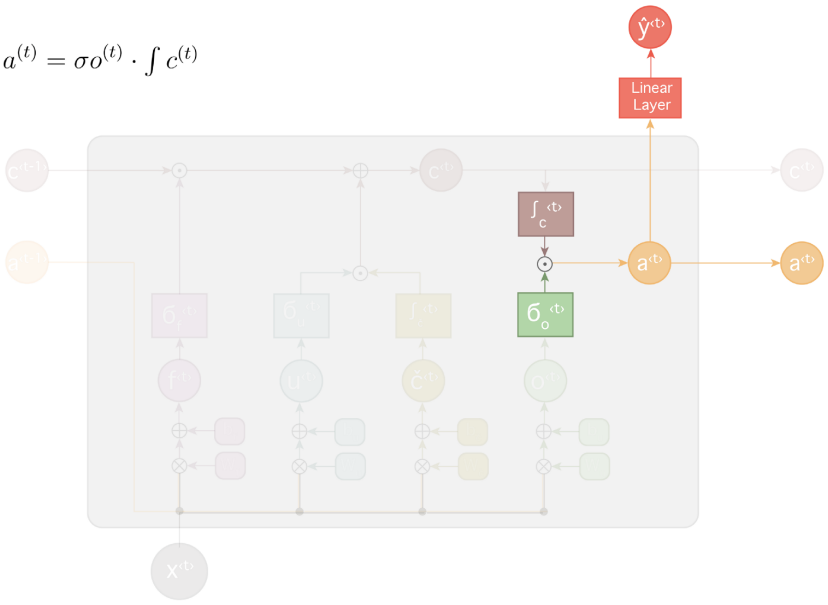
**Step 6**: Next, we have the output gate `oo<t>` which like the forget and update gates has the same linear-to-logistic computational form, and determines whether or not the memory activation `∫c<t>` would be passed to the rest of the network.

$$o^{(t)} = W_o \times \begin{bmatrix} x^{(t)}, \\ a^{(t-1)} \end{bmatrix} + b_o$$
$$\sigma o^{(t)} = sigmoid(o^{(t)})$$



**Step 7:** Finally, we compute the hidden state activation `a<t>` by multiplying the output gate `oo<t>` with the memory activation `∫c<t>` and pass on the output to the next time step and to the predicted output linear layer.

$$a^{(t)} = \sigma o^{(t)} \cdot \int c^{(t)}$$




## Conclusion

The steps above basically cover all that happens during forward propagation in a single LSTM unit. In the second part of this post we would look at deriving the LSTM gradients for back propagation and a practical implementation of the model in PyTorch.