

# LSTM as a layer

During one time step  $t$ , the LSTM

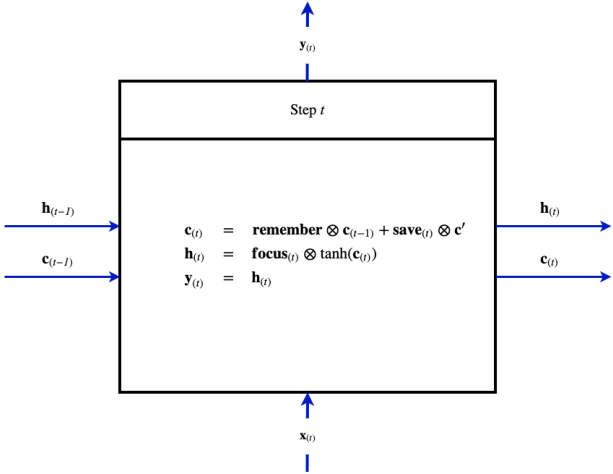
- Takes input element  $\mathbf{x}_{(t)}$
- Updates long term memory  $\mathbf{c}_{(t)}$
- Updates control state  $\mathbf{h}_{(t)}$
- Optionally outputs  $\mathbf{y}_{(t)}$

The three separate computations are functions of

- the previous short term state  $\mathbf{h}_{(t-1)}$ ,
- previous long term state  $\mathbf{c}_{(t-1)}$
- and the current input  $x_{(t)}$ .

$$\mathbf{y}_{(t)}, \mathbf{h}_{(t)}, \mathbf{c}_{(t)} = f(\mathbf{x}_{(t)}, \mathbf{h}_{(t-1)}, \mathbf{c}_{(t-1)})$$

LSTM



**remember, focus, save** are *gates*

- That control different aspects of the update process

# Conclusion

This was a high level introduction to the LSTM API.

It is similar to a vanilla RNN but separates responsibility

- For short-term transition control
- And long term memory and output
- Using an additional variable  $\mathbf{c}_{(t)}$

It will turn out that the gates are specially designed to combat the problem of vanishing/exploding gradients.

```
In [ ]: print("Done")
```