PyTorch

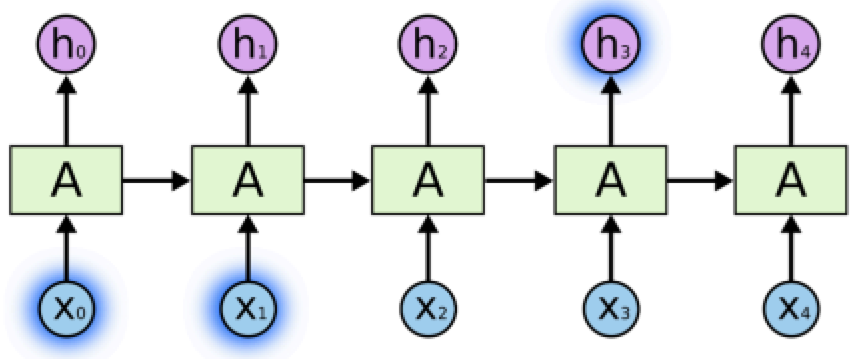
LSTM

主讲人：龙良曲

The problem of long-term dependencies

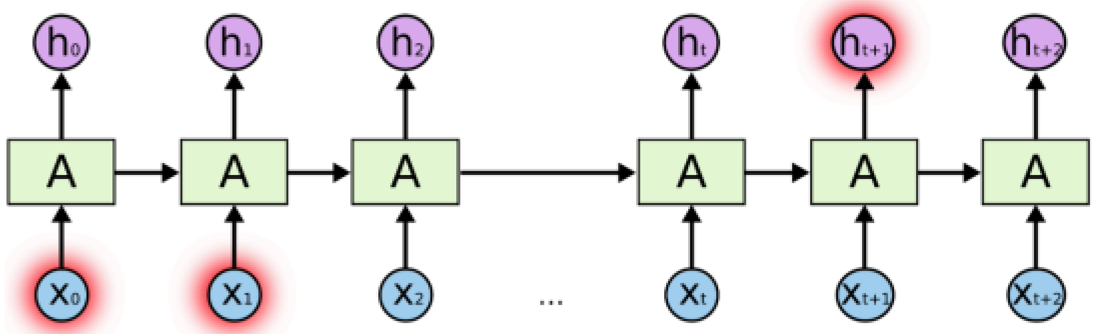
(Vanilla) RNNs connect previous information to present task:

- enough for predicting the next word for "the clouds are in the *sky*"



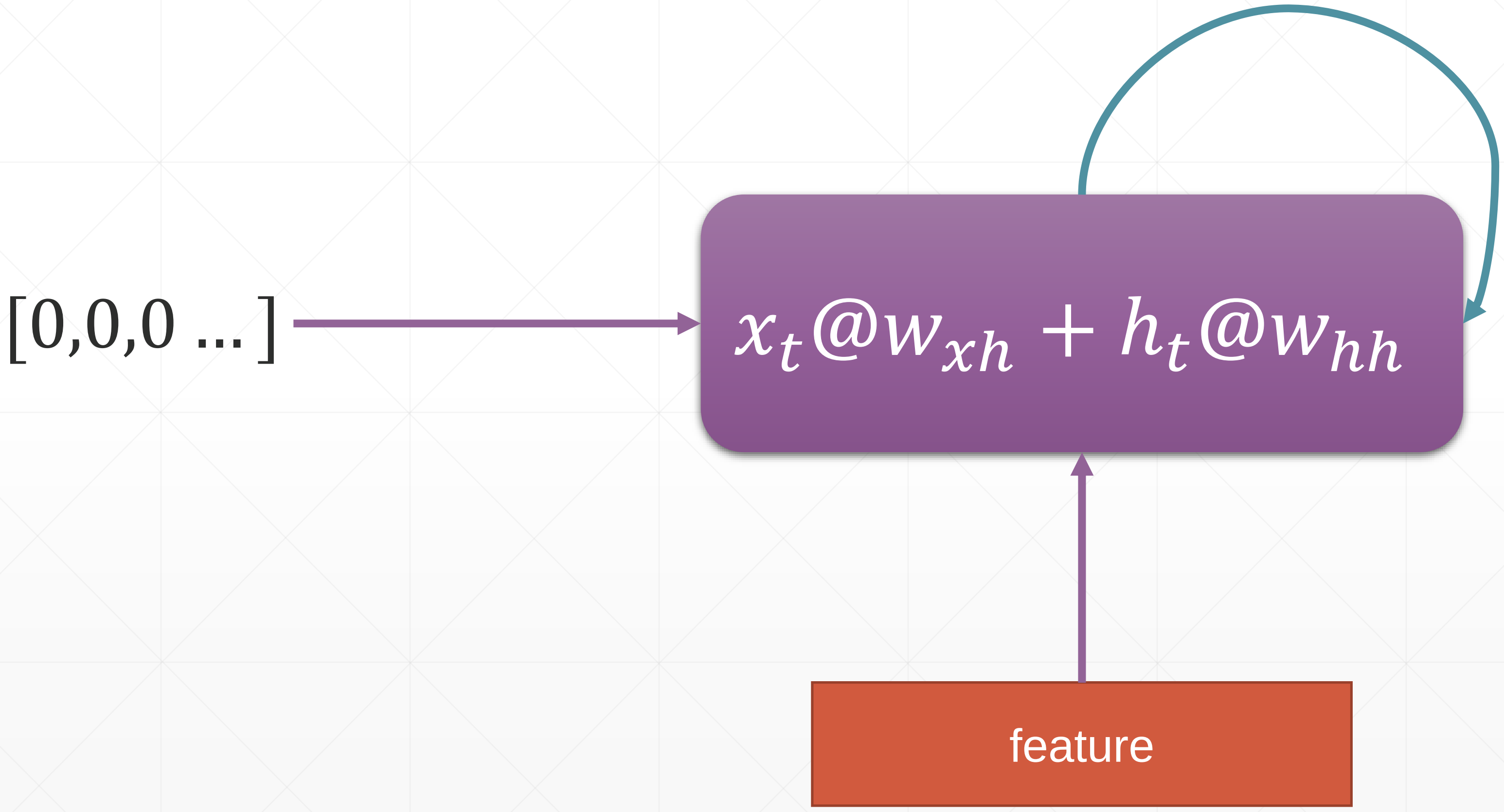
- may not be enough when more context is needed

"I grew up in France… I speak fluent *French*."

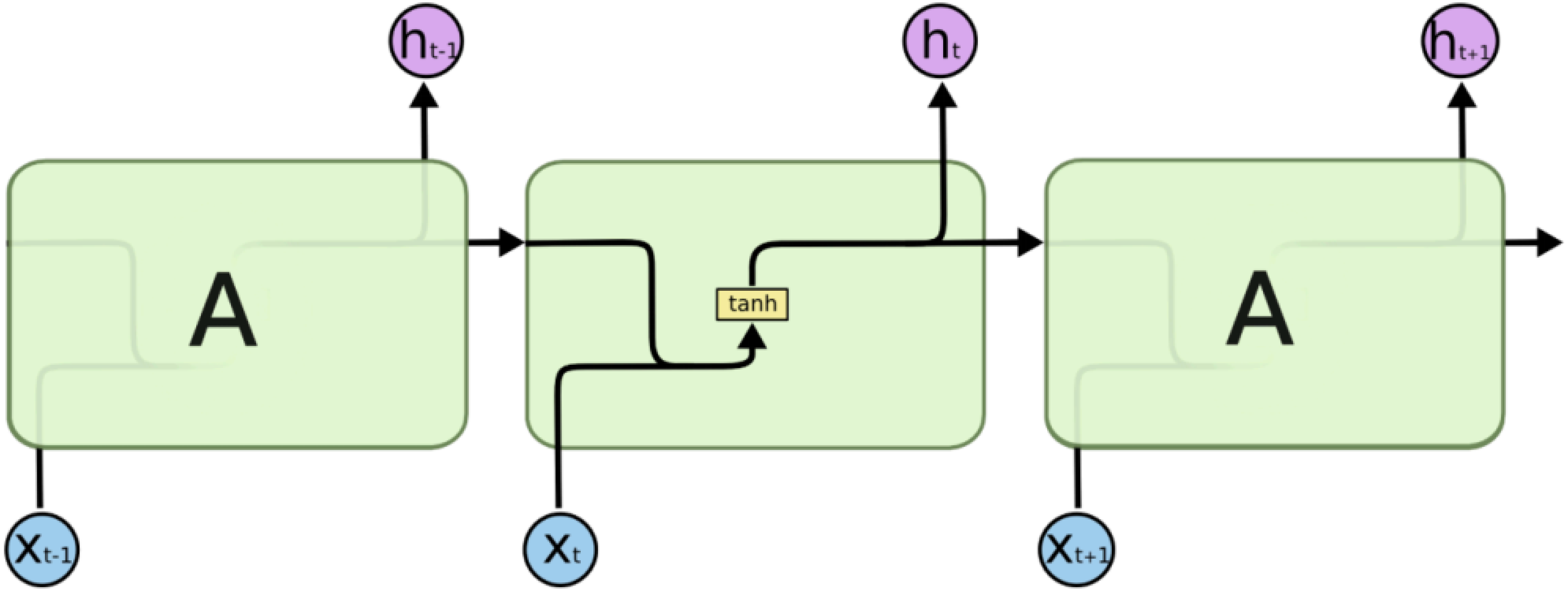


Adapted from Christopher Olah

Folded model



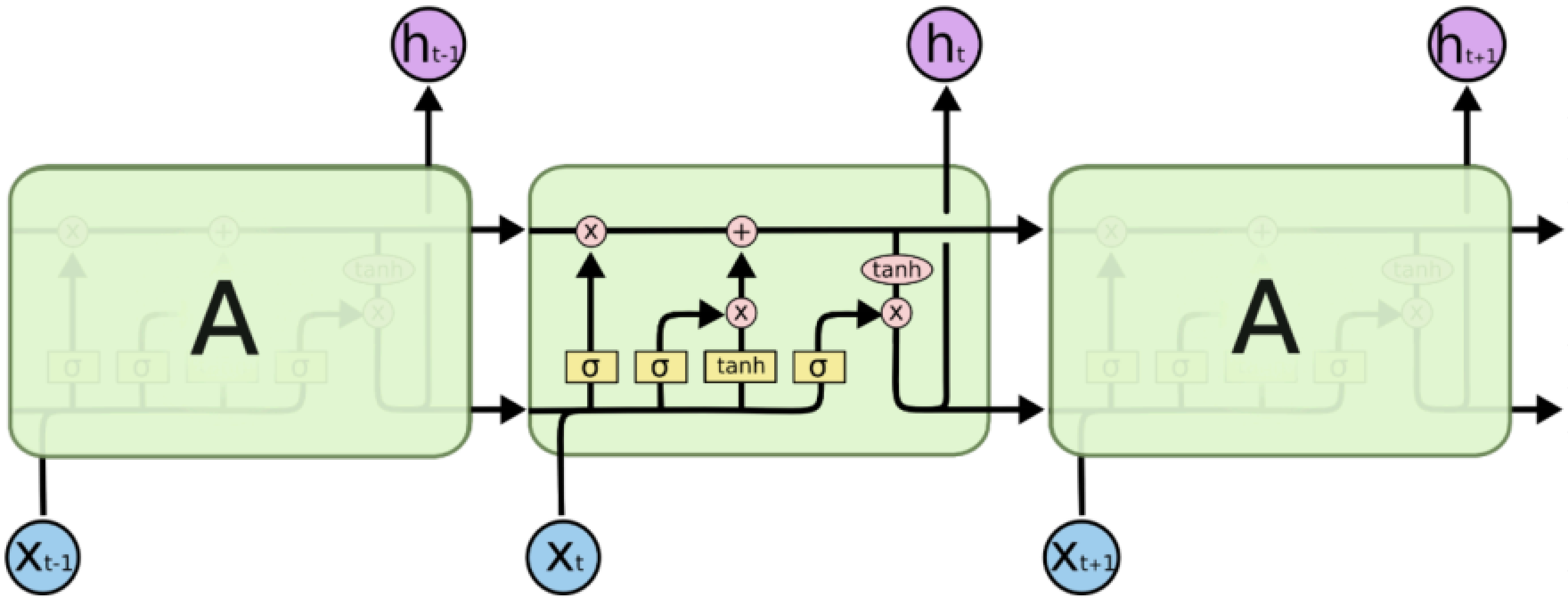
All recurrent neural networks have the form of a chain of repeating modules of neural network



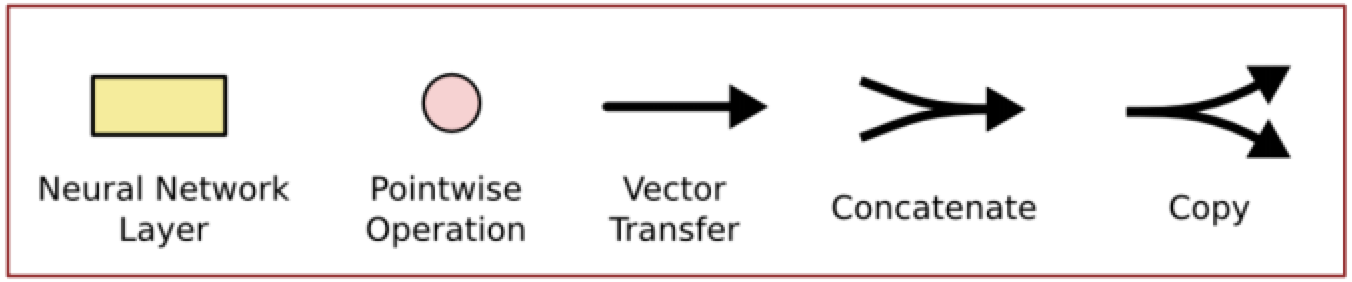
The repeating module in a standard RNN contains a single layer.

Adapted from Christopher Olah

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer there are four, interacting in a very special way.

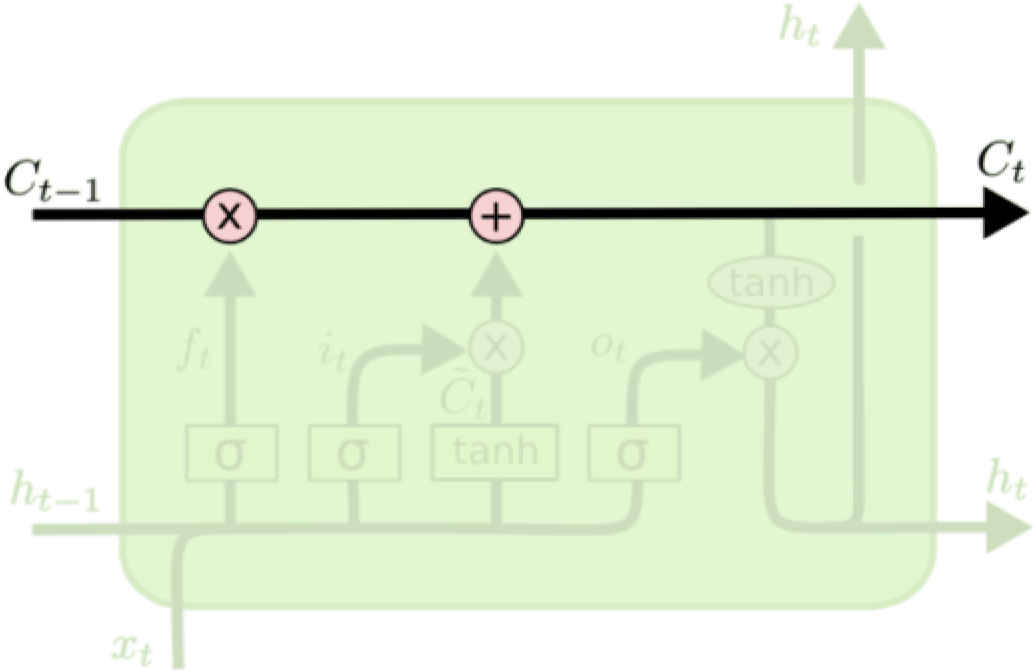


The repeating module in an LSTM contains four interacting layers.

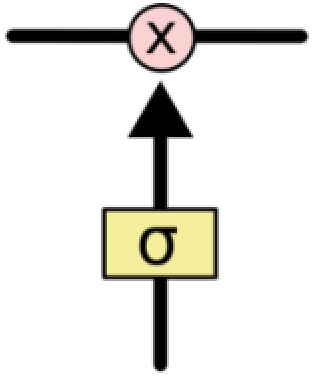


Adapted from Christopher Olah

The Core Idea Behind LSTMs : Cell State



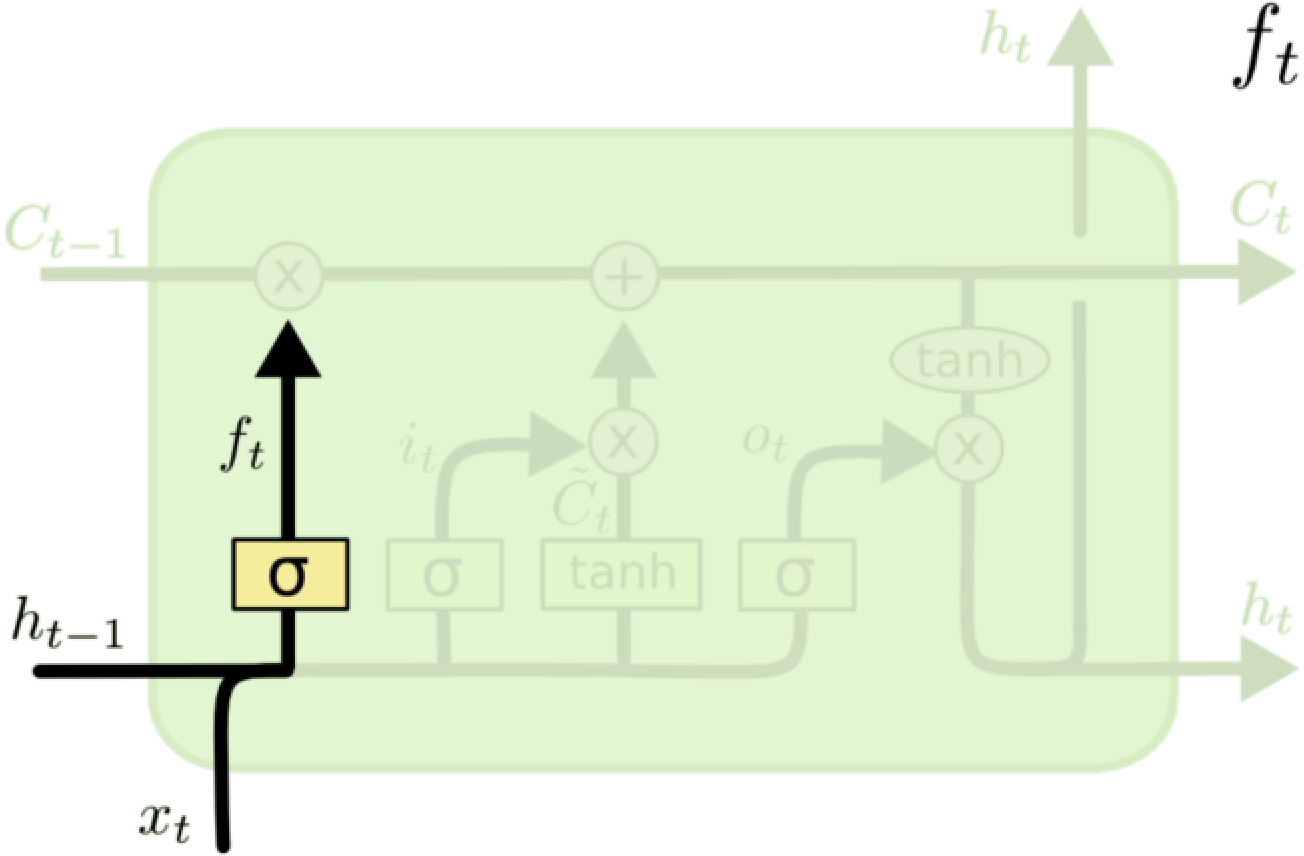
Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



An LSTM has three of these gates, to protect and control the cell state.

Adapted from Christopher Olah

LSTM : Forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

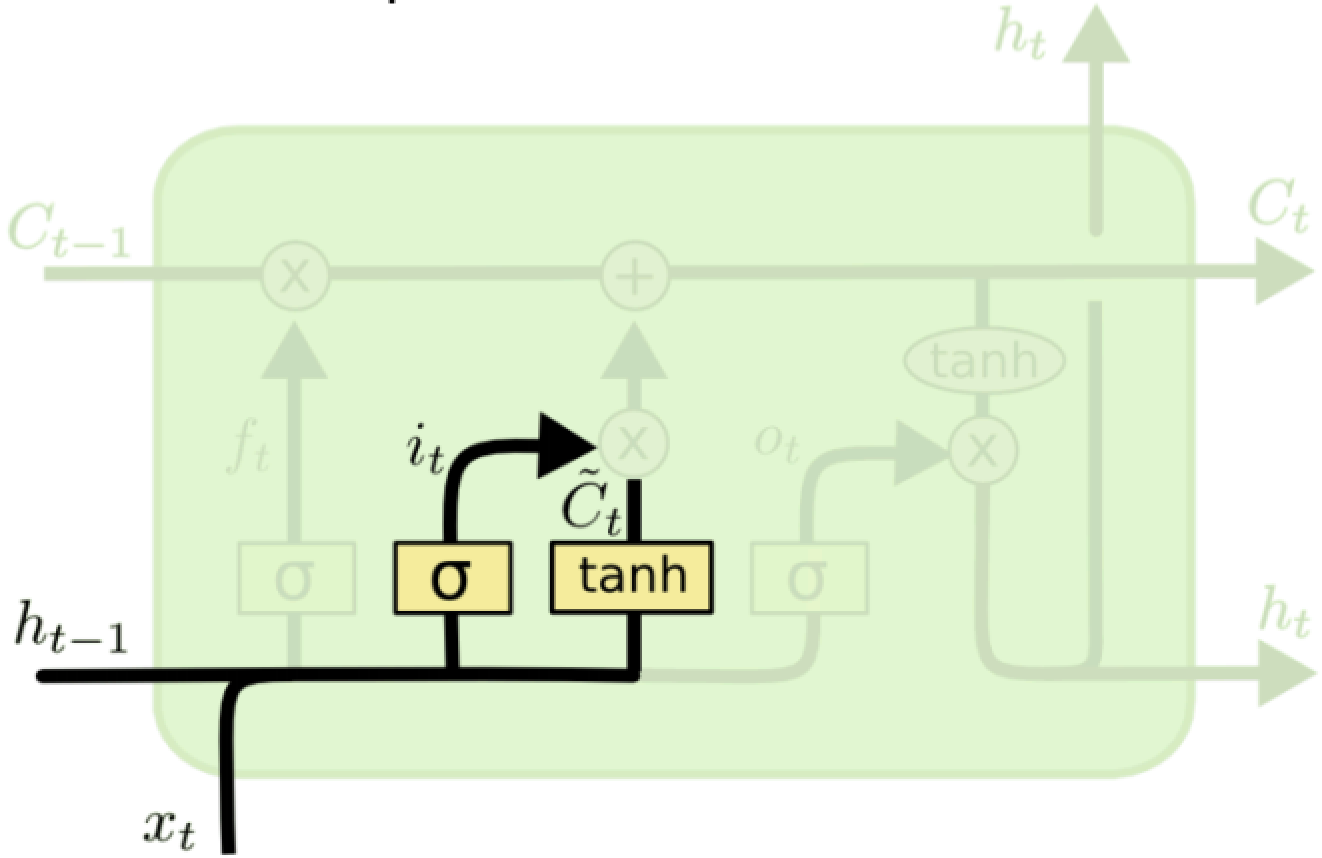
It looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}.

A 1 represents “completely keep this” while a 0 represents “completely get rid of this”.

Adapted from Christopher Olah

LSTM : Input gate and Cell State

The next step is to decide what new information we're going to store in the cell state.



a sigmoid layer called the "**input gate layer**" decides which values we'll update.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

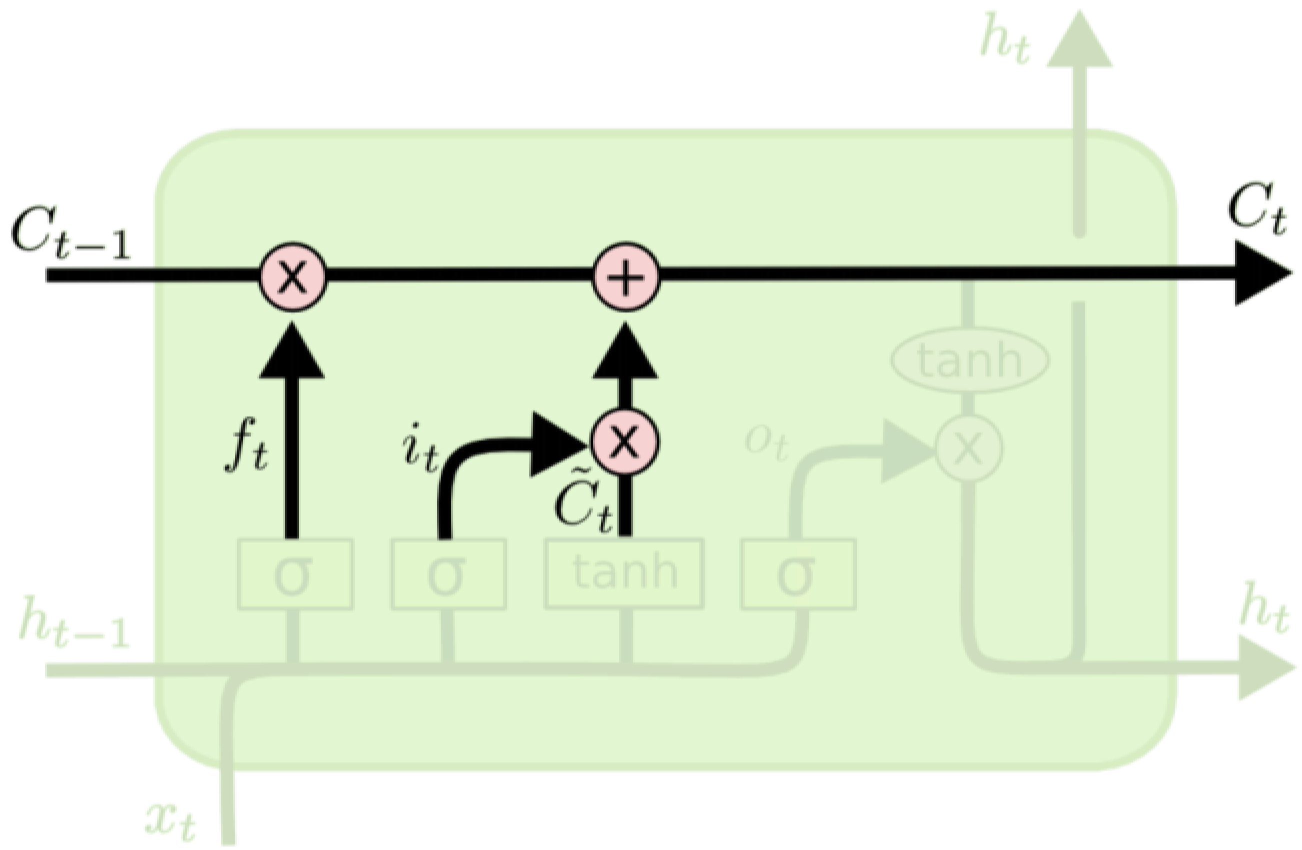
a tanh layer creates a vector of new candidate values, that could be added to the state.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Adapted from Christopher Olah

LSTM : Input gate and Cell State

It's now time to update the old cell state into the new cell state.



$$C_t = \boxed{f_t * C_{t-1}} + \boxed{i_t * \tilde{C}_t}$$

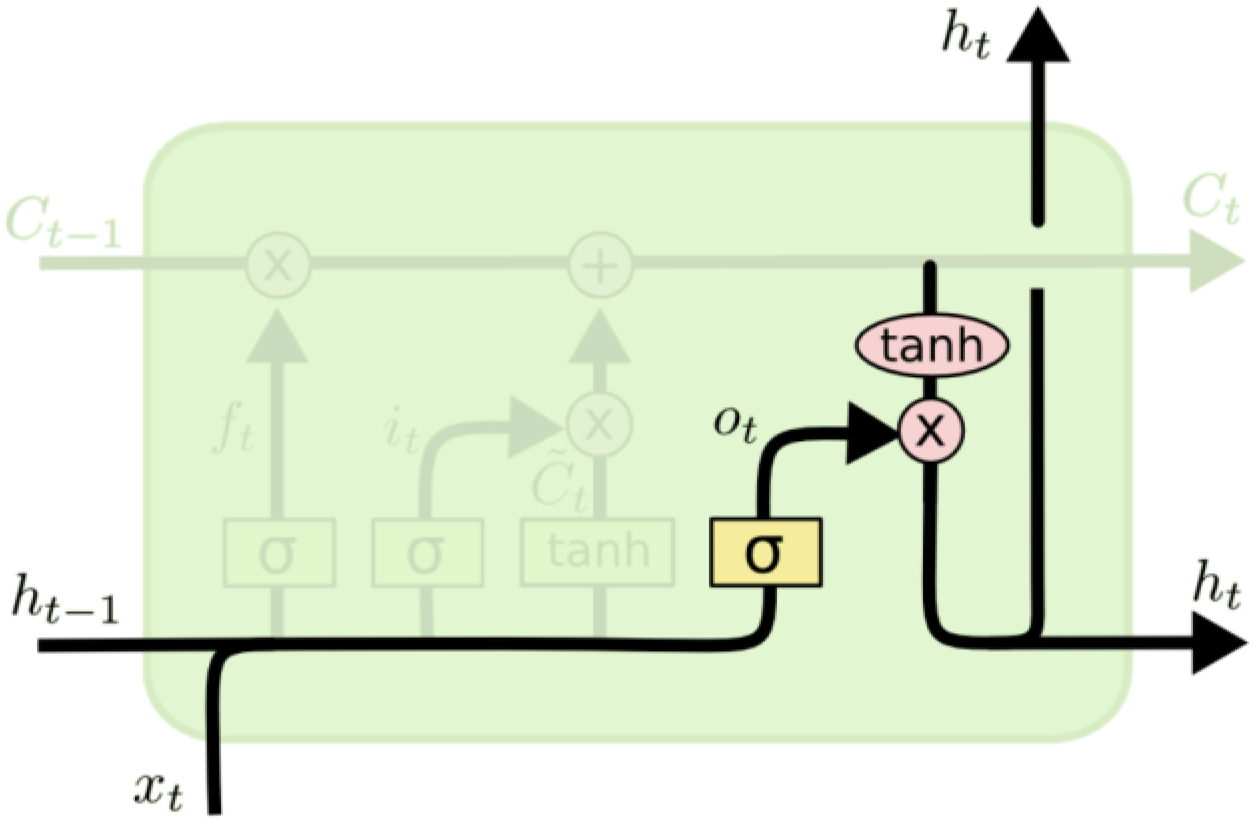
We multiply the old state by ft forgetting the things we decided to forget earlier.

Then, we add the new candidate values, scaled by how much we decided to update each state value.

Adapted from Christopher Olah

LSTM : Output

Finally, we need to decide what we're going to output.



First, we run a sigmoid layer which decides what parts of the cell state we're going to output.

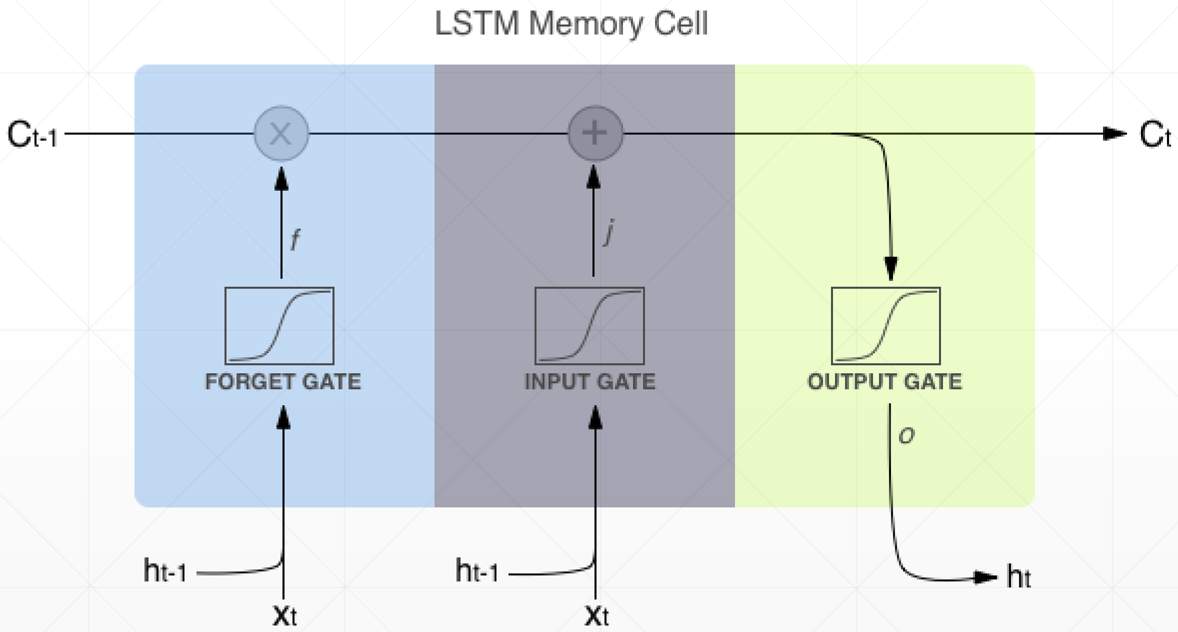
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

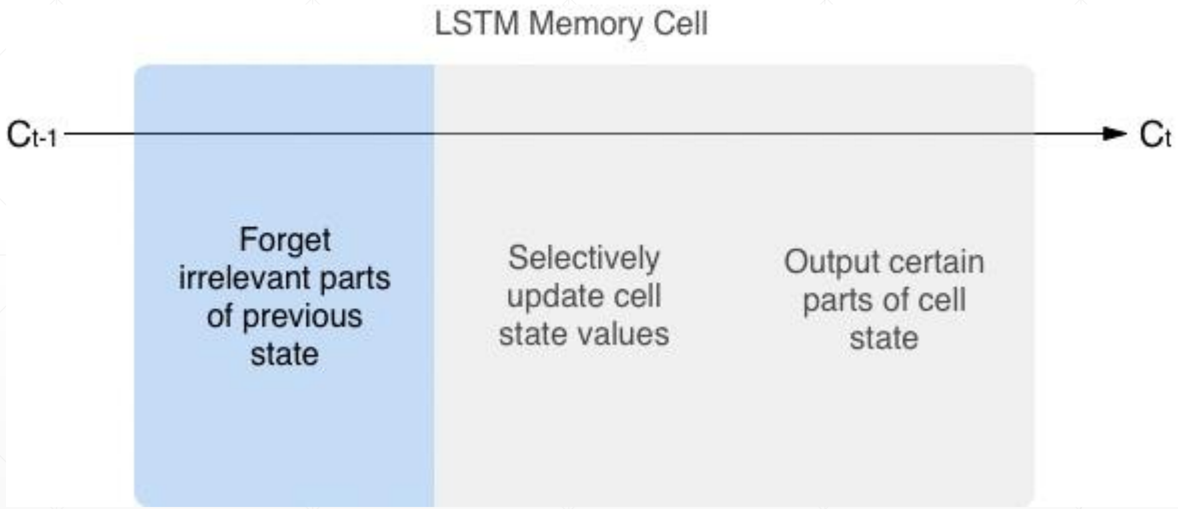
Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

$$h_t = o_t * \tanh\left(C_t\right)$$

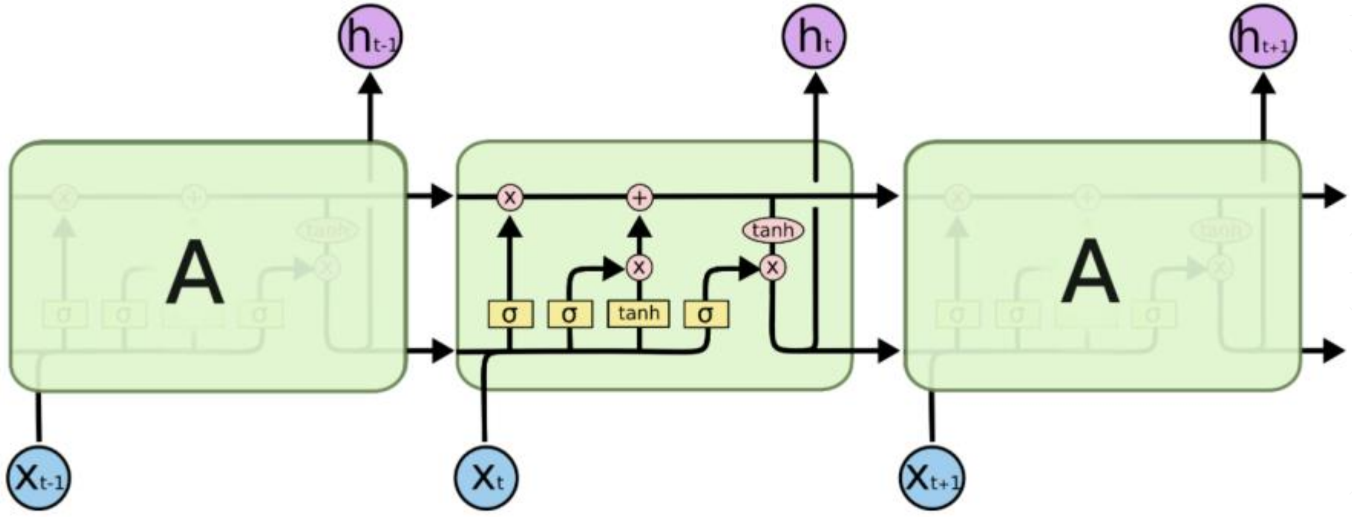
Adapted from Christopher Olah

Intuitive Pipeline





http://harinisuresh.com/2016/10/09/lstms/

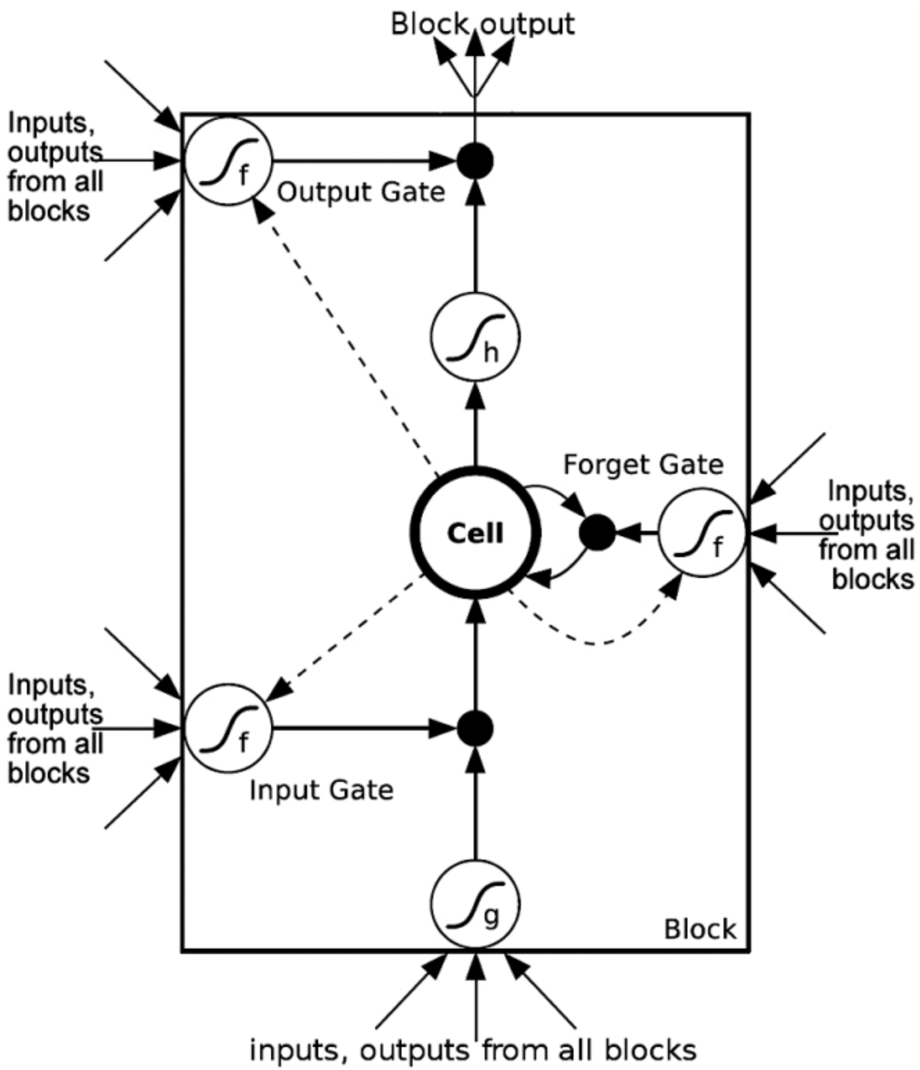


$$\begin{pmatrix} \mathbf{i}^{(t)} \\ \mathbf{f}^{(t)} \\ \mathbf{o}^{(t)} \\ \tilde{C} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix} \tag{6}$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{C} \tag{7}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh(\mathbf{c}^{(t)}). \tag{8}$$

http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%
20and%20Vanishing%20Gradients.pdf

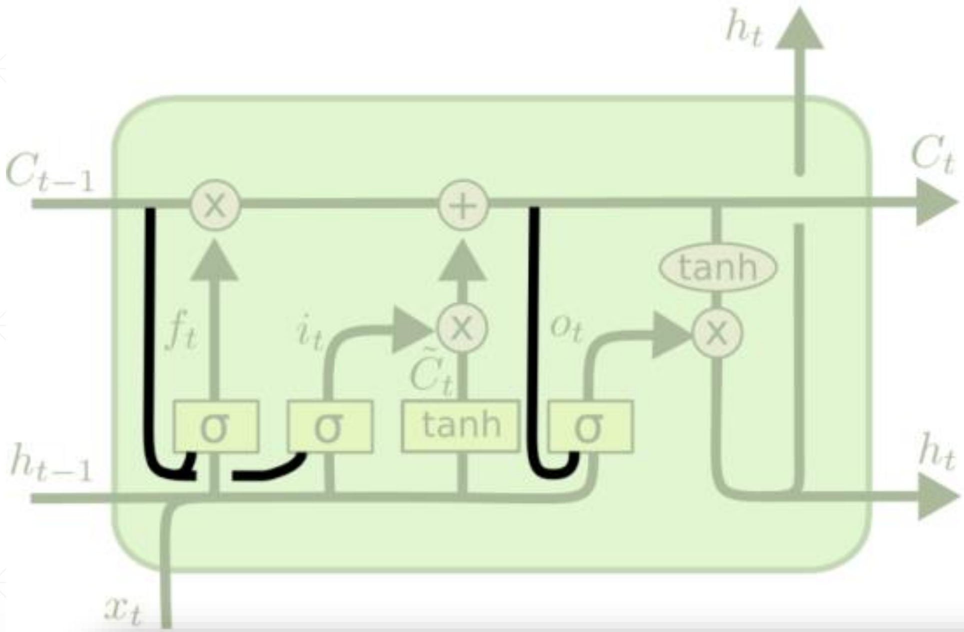


input gate	forget gate	behavior
0	1	remember the previous value
1	1	add to the previous value
0	0	erase the value
1	0	overwrite the value

How to solve Gradient Vanishing?

$$\frac{\partial C_t}{\partial C_{t-1}} = \frac{\partial C_t}{\partial f_t}\frac{\partial f_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial i_t}\frac{\partial i_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial \tilde{C}_t}\frac{\partial \tilde{C}_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial C_{t-1}}$$

$$\begin{aligned}\frac{\partial C_t}{\partial C_{t-1}} &= C_{t-1}\sigma'(\cdot)W_f * o_{t-1}tanh'(C_{t-1}) \\ &+ \tilde{C}_t\sigma'(\cdot)W_i * o_{t-1}tanh'(C_{t-1}) \\ &+ i_t \tanh'(\cdot)W_C * o_{t-1}tanh'(C_{t-1}) \\ &+ f_t\end{aligned}$$



$$\frac{\partial h_{k+1}}{\partial h_k} = diag(f'(W_I x_i + W_R h_{i-1}))W_R$$

$$\frac{\partial h_k}{\partial h_1} = \prod_i^k diag(f'(W_I x_i + W_R h_{i-1}))W_R$$

https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html
http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

下一课时

LSTM使用

Thank You.