RNN->LSTM video notes

笔记本:	深度学习		
创建时间:	2018/12/17 20:21	**更新时间:**	2019/7/2 14:23
作者:	beyourselfwb@163.com		
URL:	https://www.youtube.com/watch?v=WCUNPb-5EYI		

https://www.youtube.com/watch?v=WCUNPb-5EYI

深度好文:

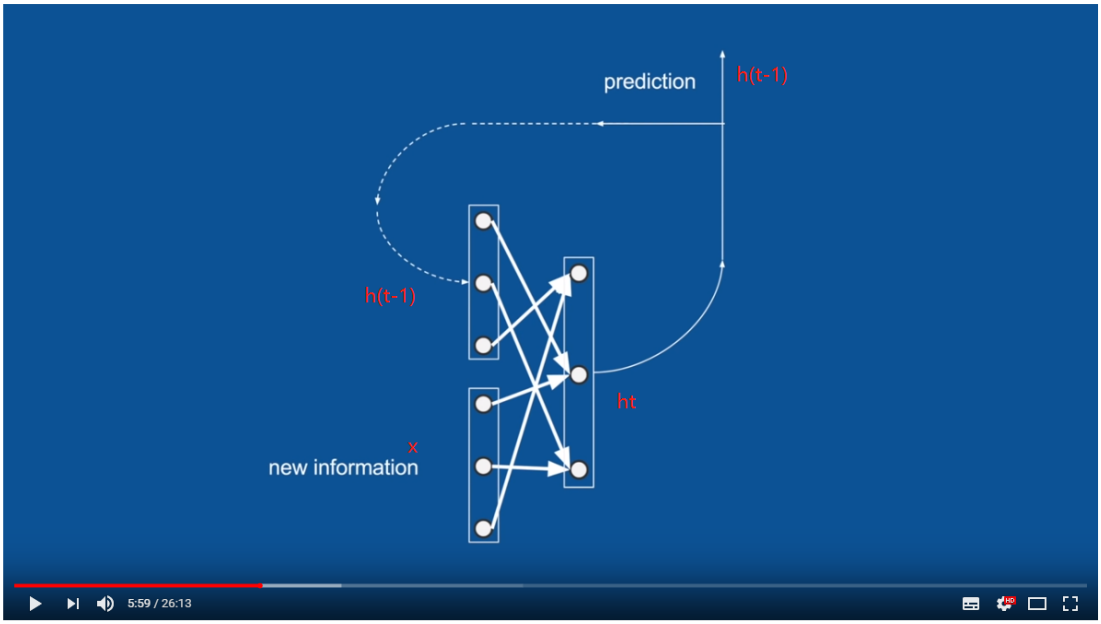
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

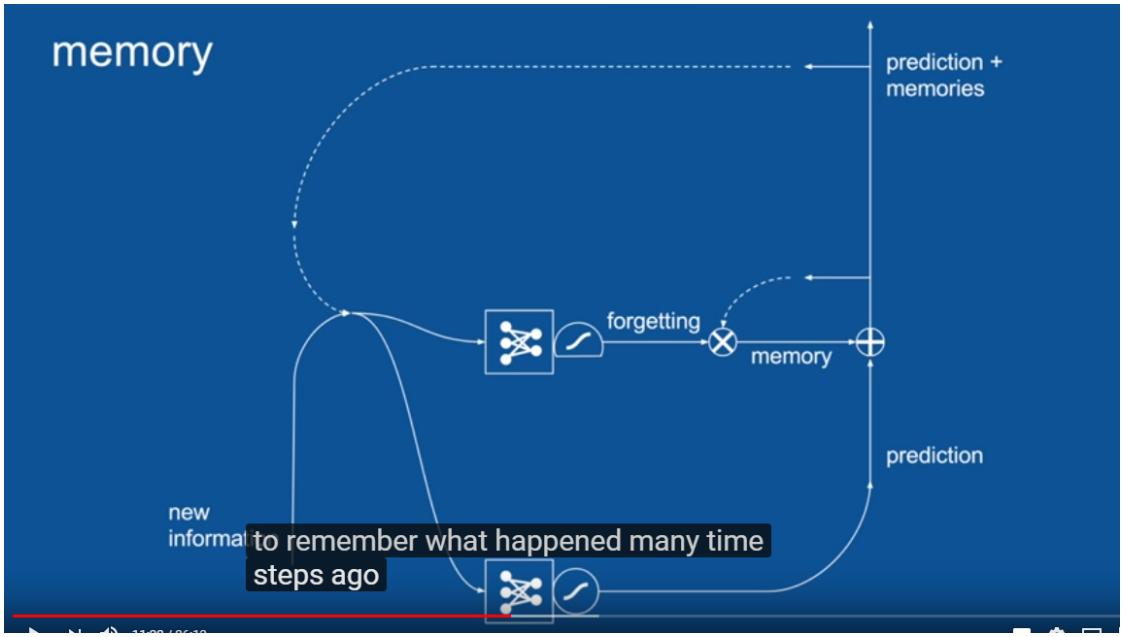
LSTM VS RNN : solve gradient vanishing/explosing, handling long-term dependencies
LSTM VS CNN : varying-length texts

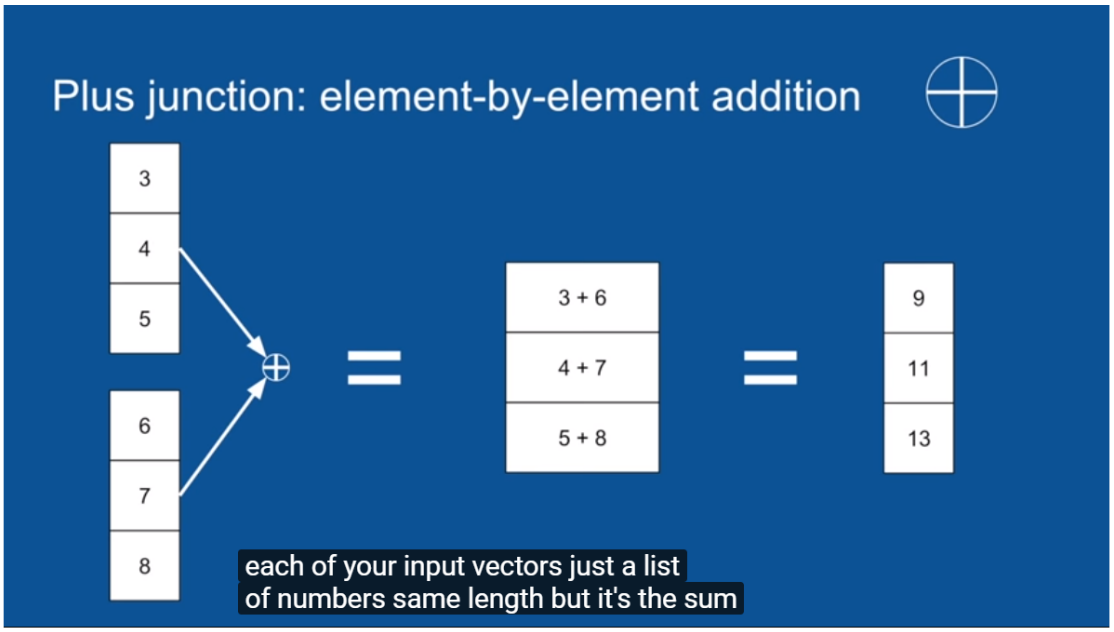
The fall of RNN / LSTM
来源：https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0
LSTM最多也就能记住 100s内的数据，1000s、10000s搞不定!
LSTM训练很耗时耗资源、也没办法运行在浏览器、手机、智能音响上

Q: How does LSTM help prevent the vanishing (and exploding) gradient problem in a recurrent neural network?
A: Use gating(pass or block, or in other words, 1 or 0) function, not activation function. And train the ‘combination’ of all those gates. Doing this, no matter how ‘deep’ your network is, or how ‘long’ the input sequence is, the network can remember those values, as long as those gates are all 1 along the path. <- This is how LSTM/GRU did the job.
from: https://www.quora.com/How-does-LSTM-help-prevent-the-vanishing-and-exploding-gradient-problem-in-a-recurrent-neural-network

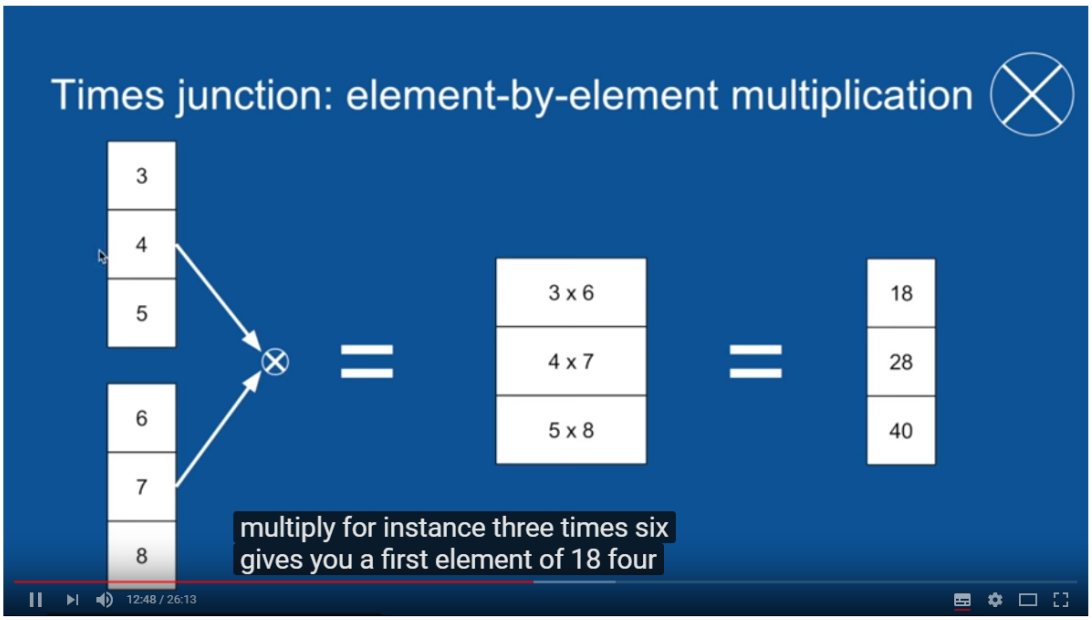
Recurrent Neural Networks (RNN) and Long Short-Term


prediction
h(t-1)
h(t-1)
ht
x
new information
5:59 / 26:13

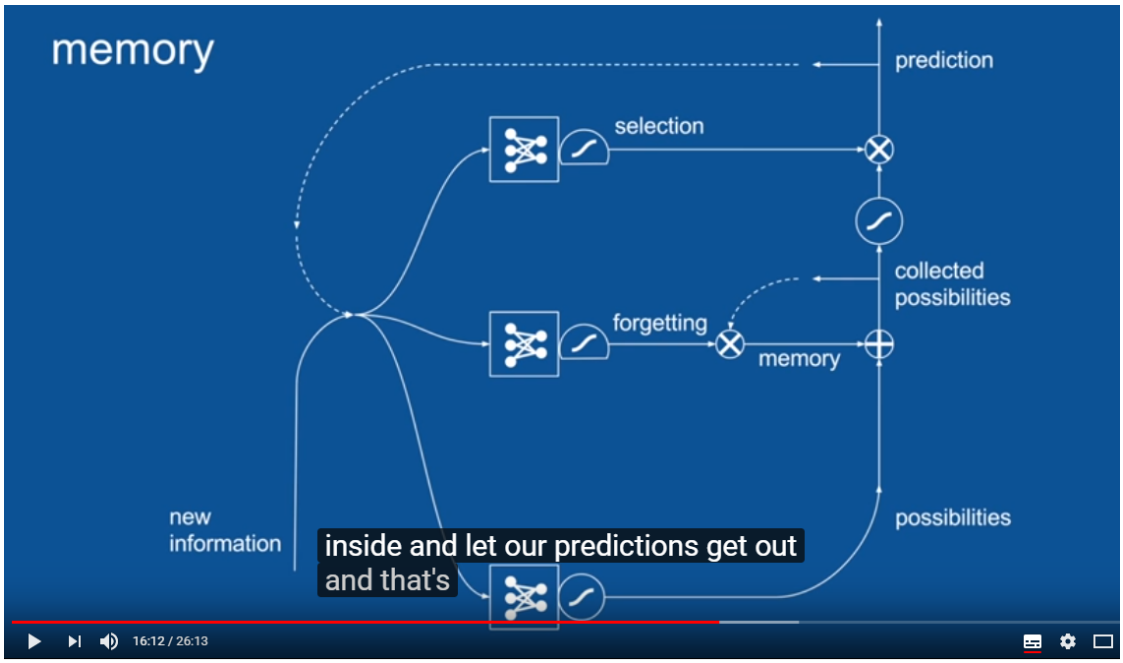

memory
prediction +
memories
forgetting
memory
prediction
new
informa
to remember what happened many time
steps ago

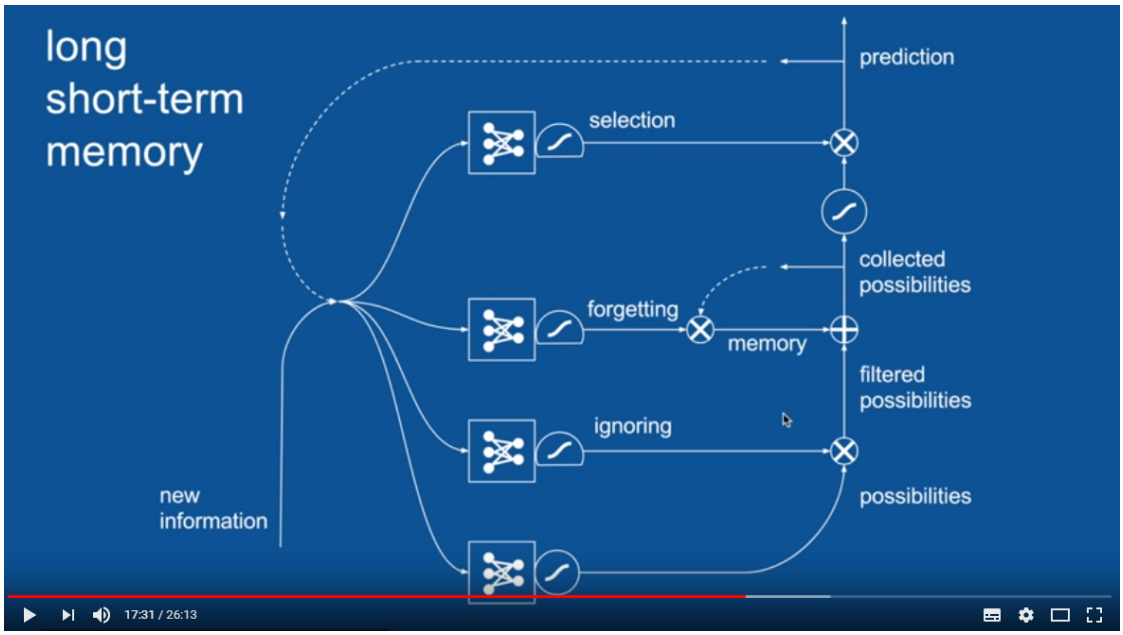


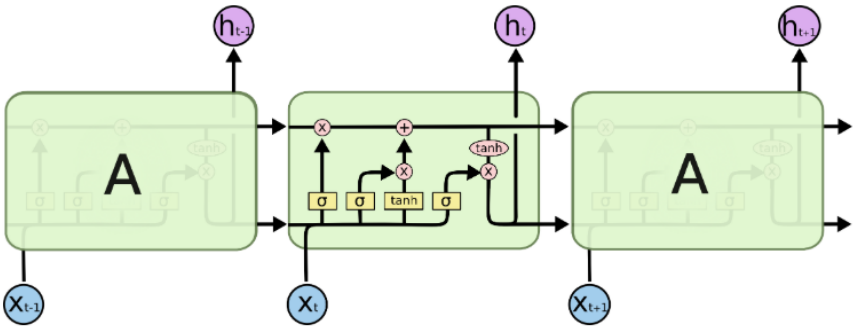
Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)



Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)

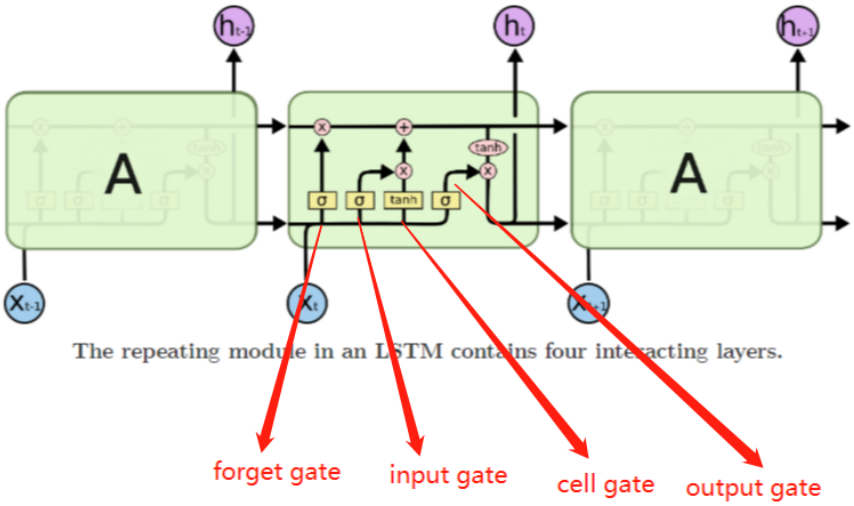






The repeating module in an LSTM contains four interacting layers.

放到一起对比



The repeating module in an LSTM contains four interacting layers.

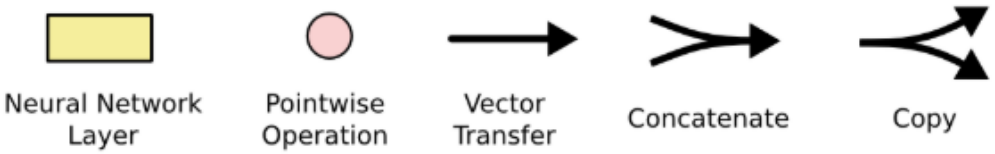
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i)$$

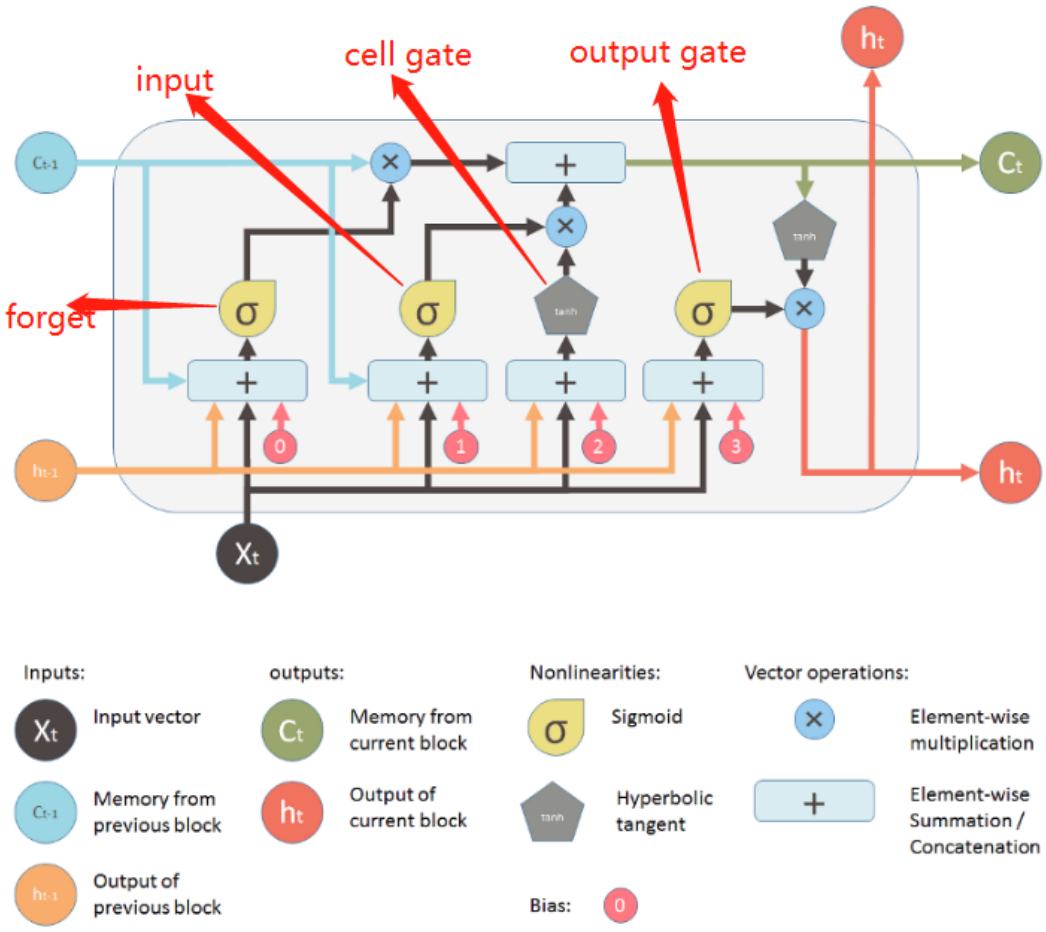
$$f_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{c}_t = f_t \odot c_{t-1} + i_t \odot \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t)$$





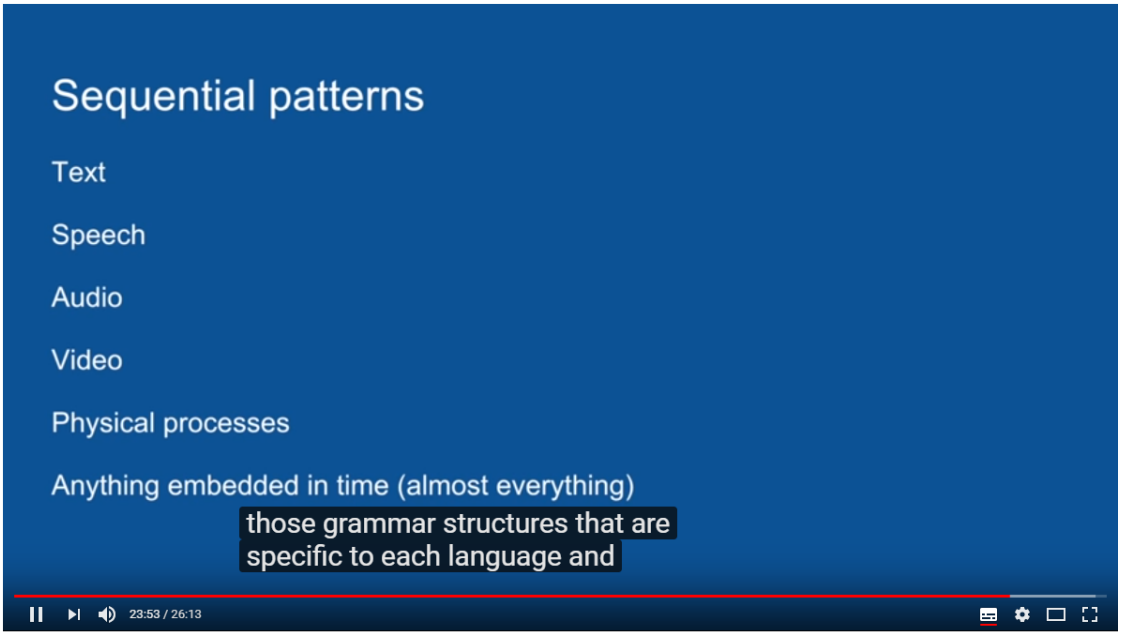
class `torch.nn.LSTMCell`(*input_size, hidden_size, bias=True*) [source]

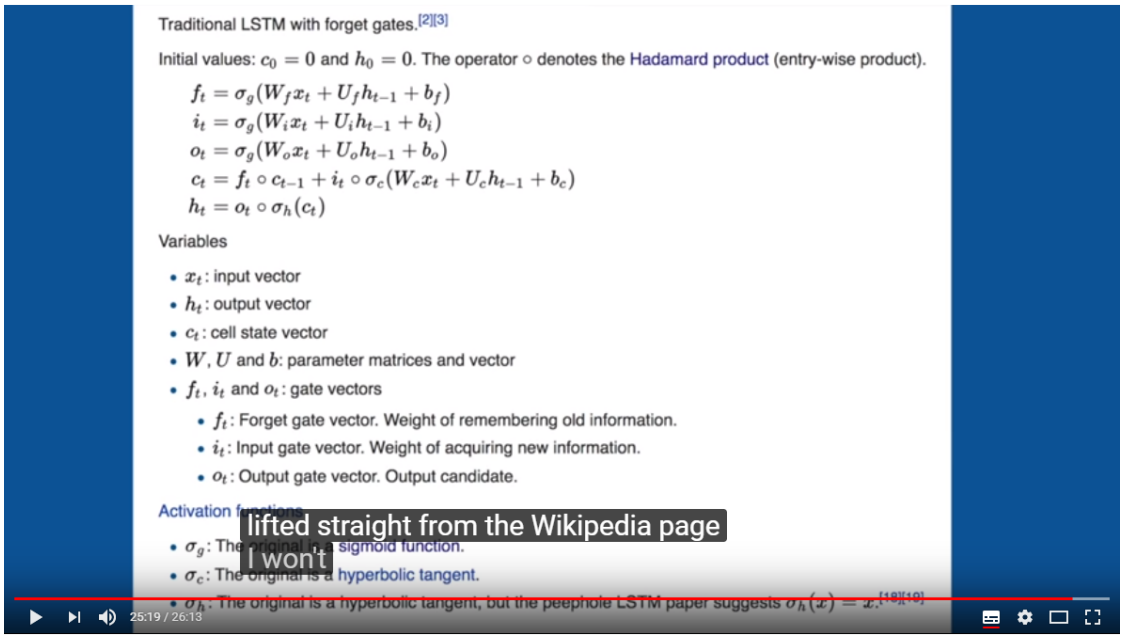
A long short-term memory (LSTM) cell.

$$
\begin{aligned}
i &= \text{sigmoid}(W_{ii}x + b_{ii} + W_{hi}h + b_{hi}) \\
f &= \text{sigmoid}(W_{if}x + b_{if} + W_{hf}h + b_{hf}) \\
g &= \tanh(W_{ig}x + b_{ig} + W_{hc}h + b_{hg}) \\
o &= \text{sigmoid}(W_{io}x + b_{io} + W_{ho}h + b_{ho}) \\
c' &= f * c + i * g \\
h' &= o * \tanh(c')
\end{aligned}
$$

Parameters:

- **input_size** – The number of expected features in the input x
- **hidden_size** – The number of features in the hidden state h
- **bias** – If *False*, then the layer does not use bias weights *b_ih* and *b_hh*. Default: `True`





Variants [edit]

In the equations below, the lowercase variables represent vectors. Matrices W_q and U_q contain, respectively, the weights of the input and recurrent connections, where q can either be the input gate i, output gate o, the forget gate f or the memory cell c, depending on the activation being calculated.

LSTM with a forget gate [edit]

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:[1][2]

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator ∘ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

LSTM with a forget gate [edit]

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:[1][2]

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

Variables [edit]

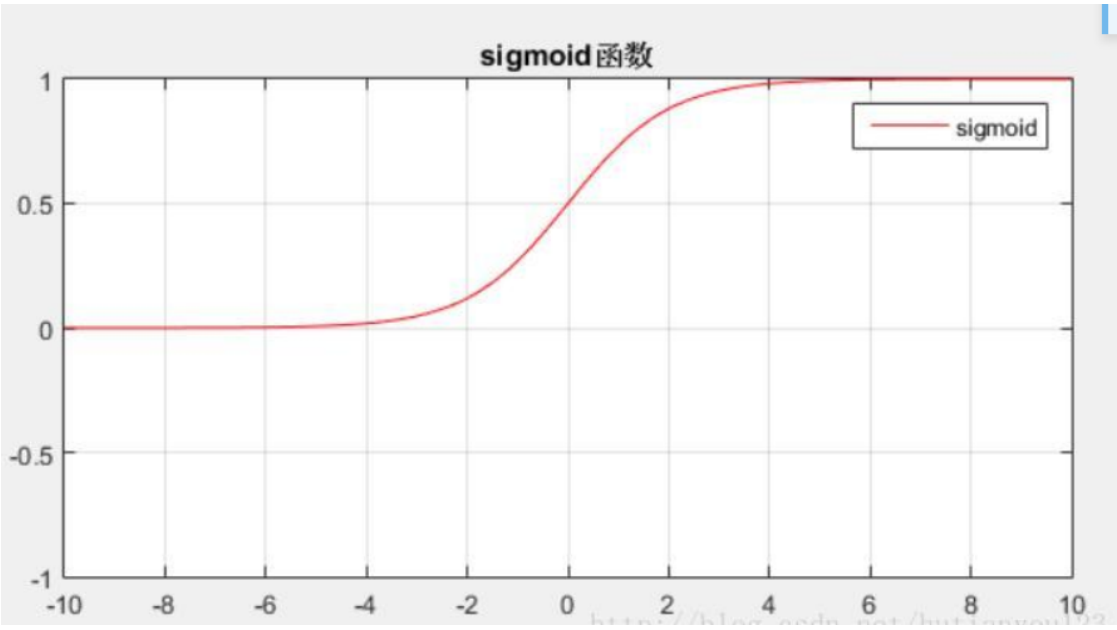
- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in \mathbb{R}^h$: forget gate's activation vector
- $i_t \in \mathbb{R}^h$: input gate's activation vector
- $o_t \in \mathbb{R}^h$: output gate's activation vector
- $h_t \in \mathbb{R}^h$: hidden state vector also known as output vector of the LSTM unit
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

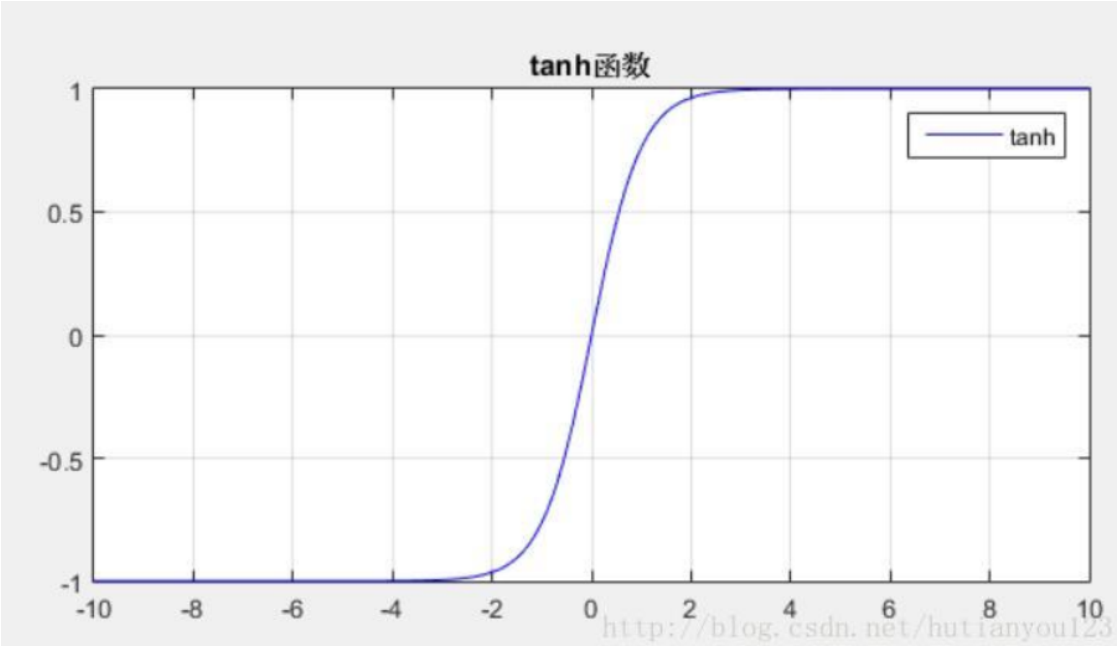
where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

Activation functions [edit]

- σ_g: sigmoid function.
- σ_c: hyperbolic tangent function.
- σ_h: hyperbolic tangent function or, as the peephole LSTM paper[*which?*] suggests, $\sigma_h(x) = x$.[20][21]

激活函数：

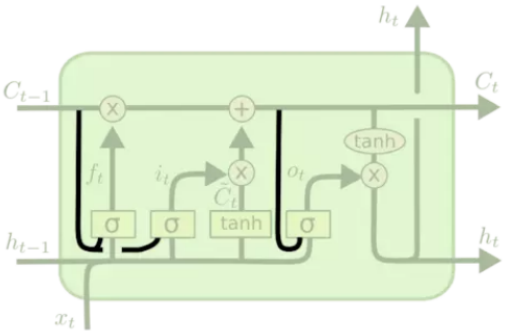




LSTM 的变体

我们到目前为止都还在介绍正常的 LSTM。但是不是所有的 LSTM 都长成一个样子的。实际上，几乎所有包含 LSTM 的论文都采用了微小的变体。差异非常小，但是也值得拿出来讲一下。

其中一个流形的 LSTM 变体，就是由 Gers & Schmidhuber (2000) 提出的，增加了“peephole connection”。是说，我们让 门层 也会接受细胞状态的输入。



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Resources

Chris Olah's tutorial

Andrej Karpathy's
- Blog post
- RNN code
- Stanford CS231n lecture

The DeepLearning 4J tutorial has some helpful discussion and a longer list of good resources.

How Neural Networks Work [video]