循环神经网络(RNN) 比CNN多了记忆细胞 CNN用于分类

层次结构

RNN描述

LSTM

循环神经网络RNN-应用场景

自然语言处理(NLP) 语言模型与文本生成

机器翻译

语音识别

图像描述生成

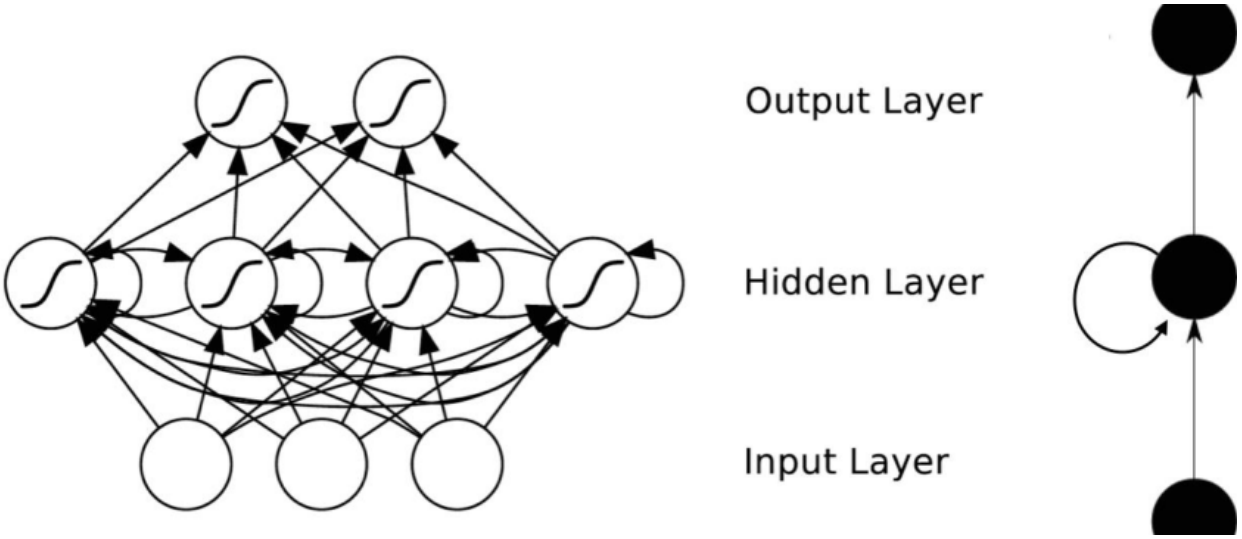
循环神经网络

为什么有BP神经网络、CNN，还需要RNN?

BP神经网络和CNN的输入输出都是互相独立的；但是实际应用中有些场景输出内容和之前的内容是有关联的。

RNN引入“记忆”的概念；循环指其每一个元素都执行相同的任务，但是输出依赖于输入和“记忆”。

循环神经网络RNN-结构



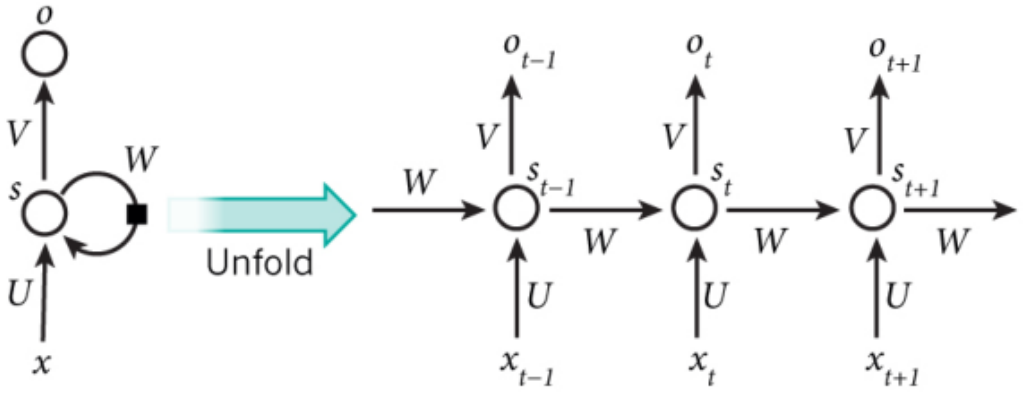
将序列按时间展开就可以得到RNN的结构

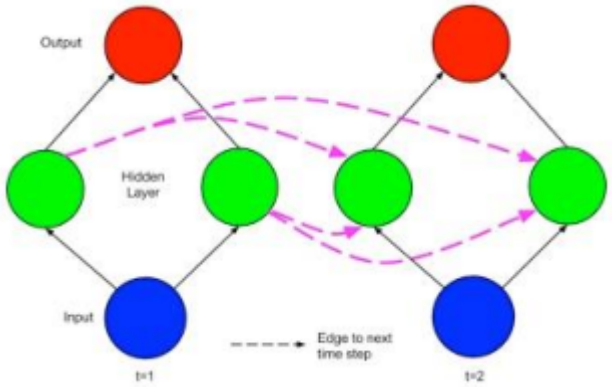
X_t 是时间t处的输入

S_t 是时间t处的“记忆”，$S_t = f(UX_t + WS_{t-1})$，U是比例

f可以是非线性转换函数，比如tanh等

O_t 是时间t处的输出，比如是预测下一个词的话，可能是softmax输出的属于每个候选词的概率，O_t =softmax(VS_t)



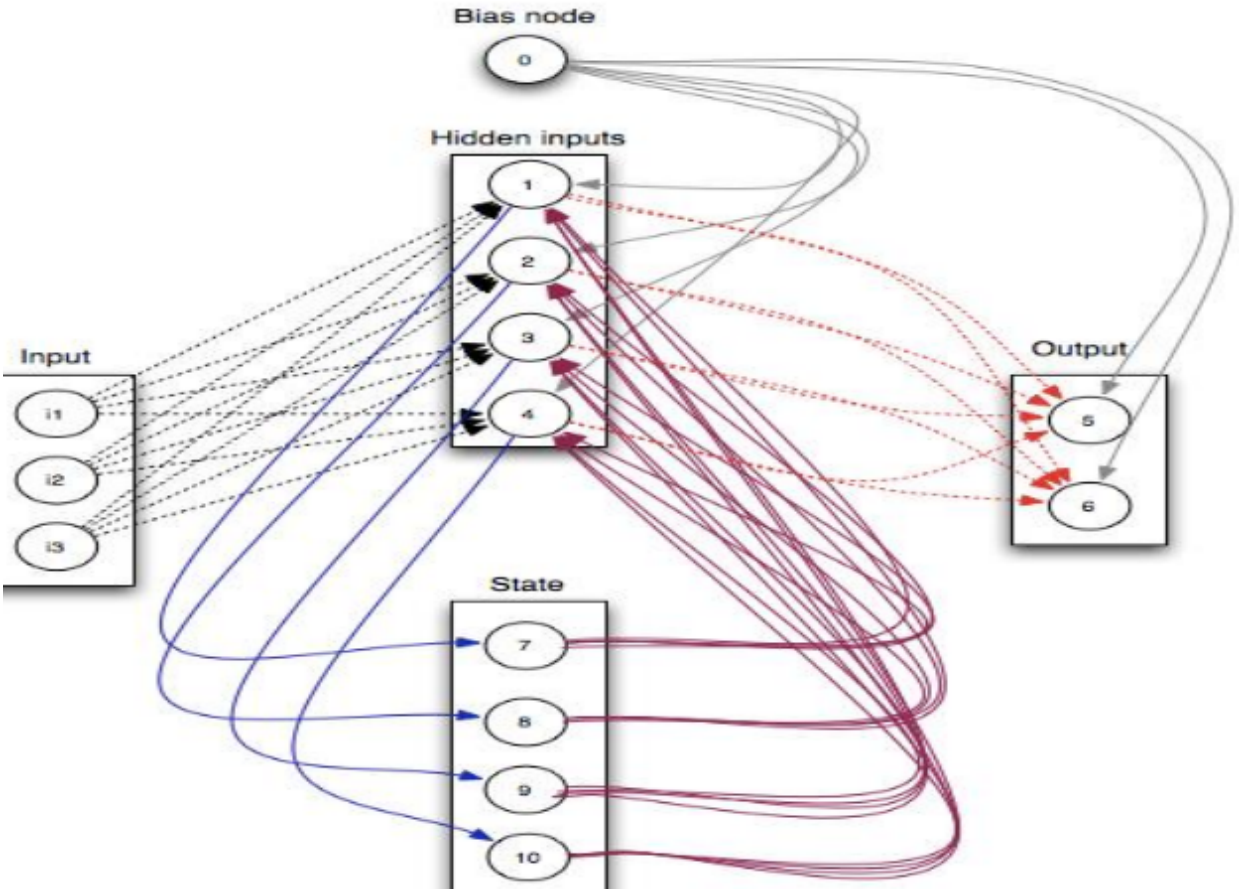


可以将隐状态S_t作为“记忆体”，捕捉之前时间节点上的信息

输出O_t的由当前时间及之前所有的“记忆”共同计算得到

不同于CNN，在RNN中，实际上整个神经网络共享一组参数(U,V,W)，这样极大

的减小了需要训练和预估的参数量

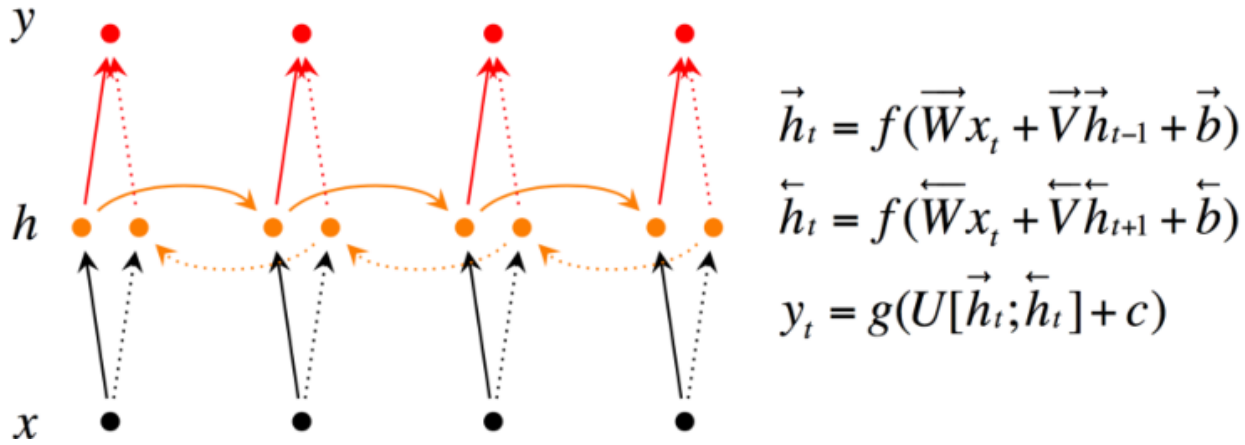


循环神经网络RNN-Bidirectional RNN

Bidirectional RNN(双向RNN)假设当前t的输出不仅仅和之前的序列有关，并且

还与之后的序列有关，例如：预测一个语句中缺失的词语那么需要根据上下文进行预测；Bidirectional RNN是一个相对简单的RNNs，由两个RNNs上下叠加在

一起组成。输出由这两个RNNs的隐藏层的状态决定。



$$\vec{h}_t = f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b})$$

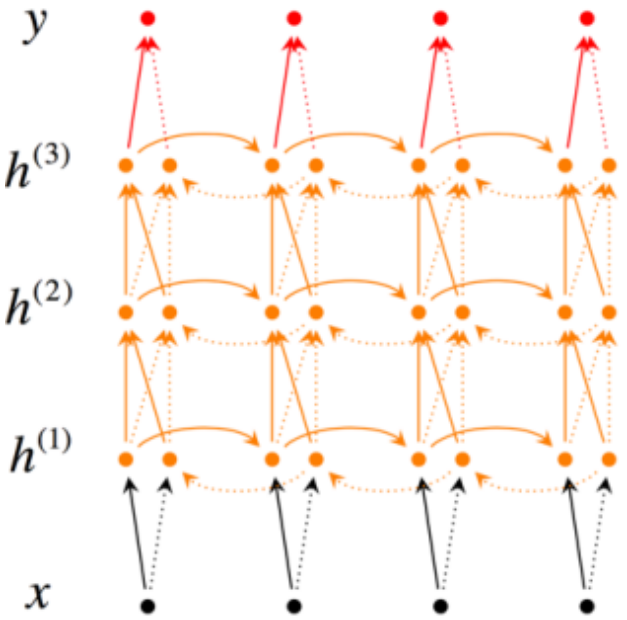
$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\vec{h}_t; \overleftarrow{h}_t] + c)$$

循环神经网络RNN-Deep(Bidirectional) RNN

Deep Bidirectional RNN(深度双向RNN)类似Bidirectional RNN，区别在于每

个每一步的输入有多层网络，这样的话该网络便具有更加强大的表达能力和学习能力，但是复杂性也提高了，同时需要训练更多的数据。



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

循环神经网络RNN-BPTT

RNN的训练和CNN/ANN训练一样，同样适用BP算法误差反向传播算法。区别在于：RNN中的参数U\V\W是共享的，并且在随机梯度下降算法中，每一步的输出不仅仅依赖当前步的网络，并且还需要前若干步网络的状态，那么这种BP改

版的算法叫做Backpropagation Through Time(BPTT)；BPTT算法和BP算法

一样，在多层训练过程中(长时依赖<即当前的输出和前面很长的一段序列有关，

一般超过10步>)，可能产生梯度消失和梯度爆炸的问题。

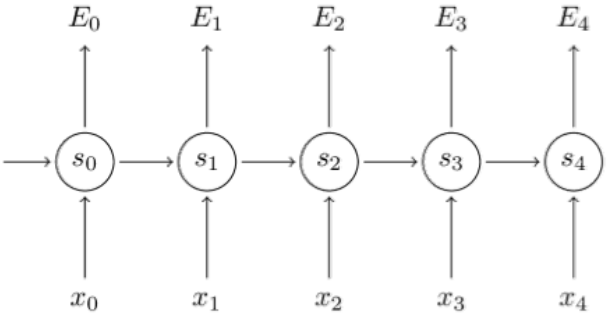
BPTT和BP算法思路一样，都是求偏导，区别在于需要考虑时间对step的影响

循环神经网络RNN-BPTT

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\hat{y}_t = \mathrm{softmax}(Vs_t)$$

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$



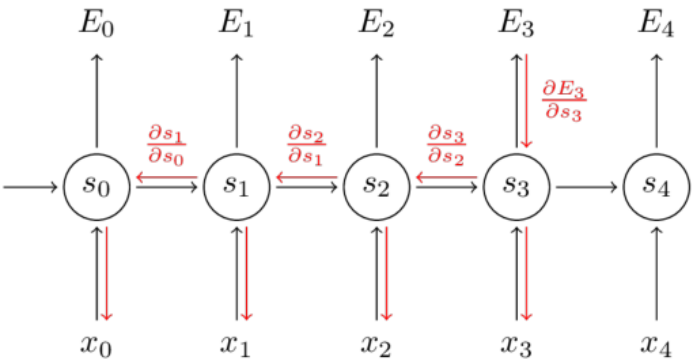
$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t \qquad \frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

链式法则

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$s_3 = \tanh(Ux_3 + Ws_2)$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$



$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$

链式法则

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\left(\prod_{j=k+1}^{3}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_k}{\partial W}$$

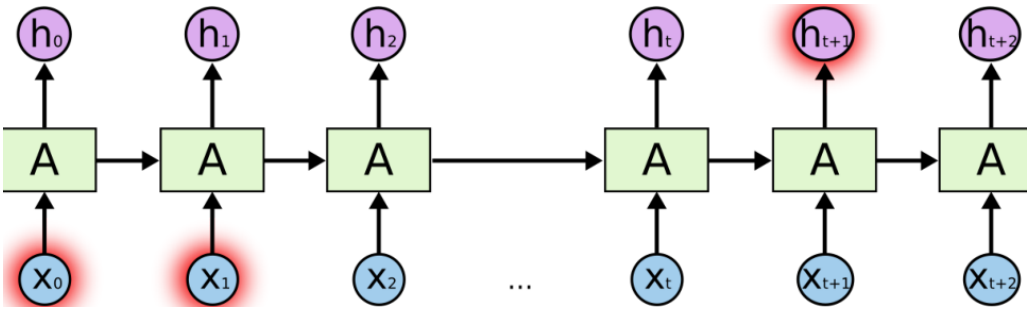
循环神经网络RNN-LSTM

在BPTT计算中，介绍到对于长期依赖的问题，没法进行解决，可能产生梯度消失和梯度爆炸的问题；LSTM特别适合解决这类需要长时间依赖的问题。

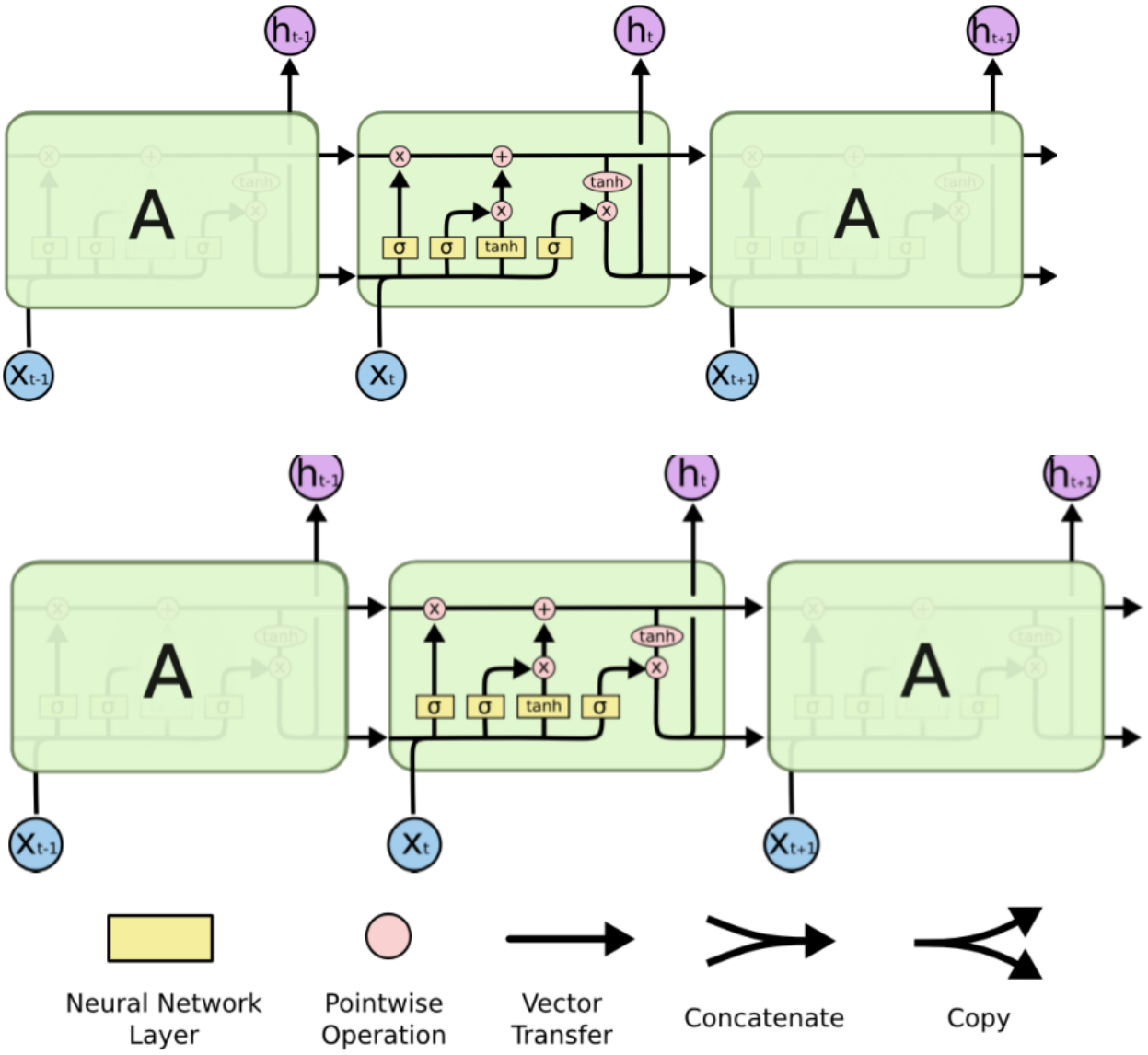
LSTM是RNN的一种，大体结构一致，区别在于：

LSTM的“记忆细胞”是改造过的

该记录的信息会一直传递，不该记录的信息会被截断掉



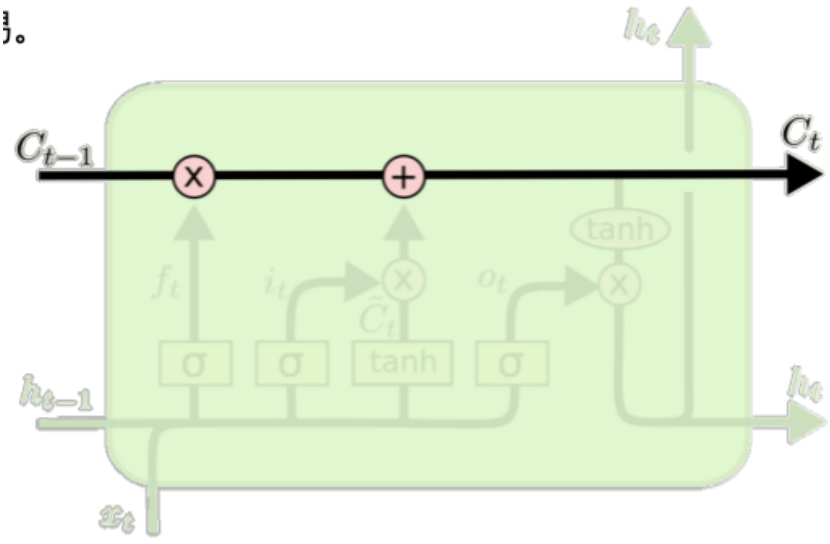
循环神经网络RNN-LSTM结构



LSTM关键：“细胞状态”

细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变很容易。

。



LSTM怎么控制“细胞状态”？

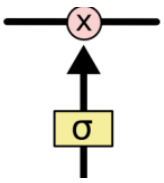
LSTM可以通过gates(“门”)结构来去除或者增加“细胞状态”的信息

包含一个sigmoid神经网络层次和一个pointwist乘法操作

Sigmoid层输出一个0到1之间的概率值，描述每个部分有多少量可以通过，0表示“不允

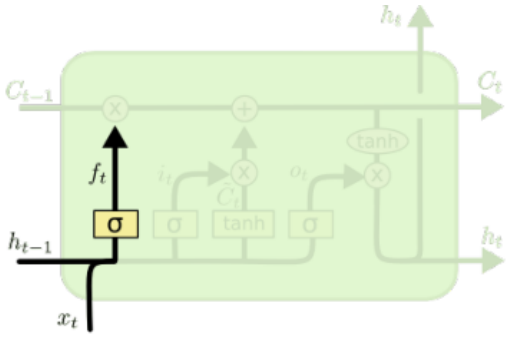
许任务变量通过”，1表示“运行所有变量通过”

LSTM中主要有三个“门”结构来控制“细胞状态”



第一个“门”==>“忘记门”：决定从“细胞状态”中丢弃什么信息；比如在语

言模型中，细胞状态可能包含了性别信息(“他”或者“她”)，当我们看到新的代名词的时候，可以考虑忘记旧的数据



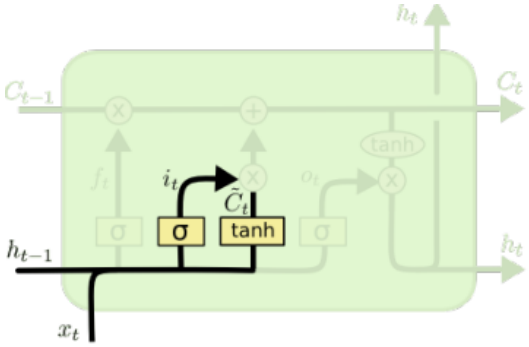
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

第二个“门”==>“信息增加门”：决定放什么新信息到“细胞状态”中；

Sigmoid层决定什么值需要更新；

Tanh层创建一个新的候选向量Ct；

主要是为了状态更新做准备



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

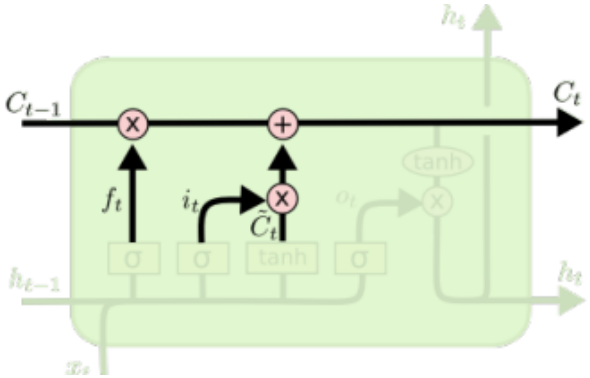
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

经过第一个和第二个“门”后，可以确定传递信息的删除和增加，即可以进行“细胞状态”的更新

更新C t-1 为C t；

将旧状态与f t 相乘，丢失掉确定不要的信息；

加上新的候选值i t *C t 得到最终更新后的“细胞状态”



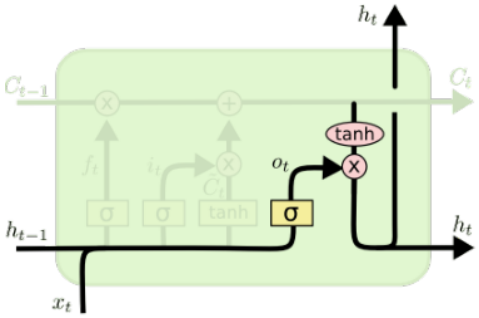
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

第三个“门”==>基于“细胞状态”得到输出；

首先运行一个sigmoid层来确定细胞状态的那个部分将输出

使用tanh处理细胞状态得到一个-1到1之间的值，再将它和sigmoid门的输出相乘，输出

程序确定输出的部分。



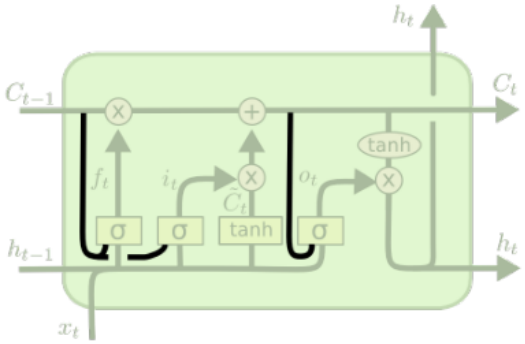
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

循环神经网络RNN-LSTM结构变种

变种1

增加“peephole connections”层

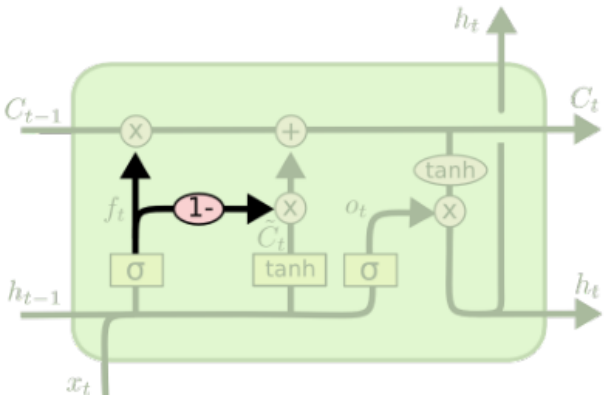
让门层也接受细胞状态的输入



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

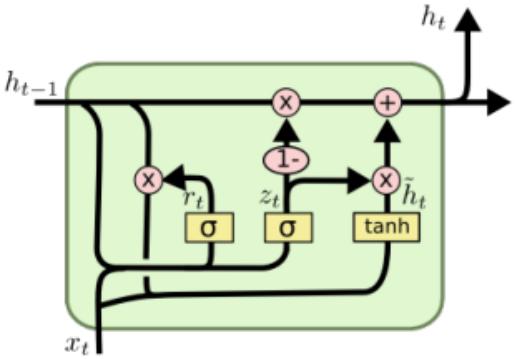
变种2

通过耦合忘记门和更新输入门(第一个和第二个门)；也就是不再单独的考虑忘记什么、增
加什么信息，而是一起进行考虑。



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Gated Recurrent Unit(GRU)，2014年提出
将忘记门和输入门合并成为一个单一的更新门
同时合并了数据单元状态和隐藏状态
结构比LSTM的结构更加简单



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$