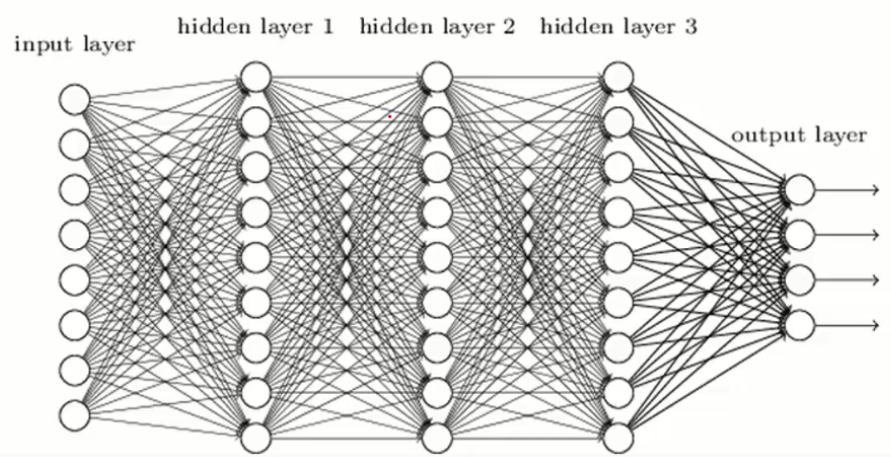
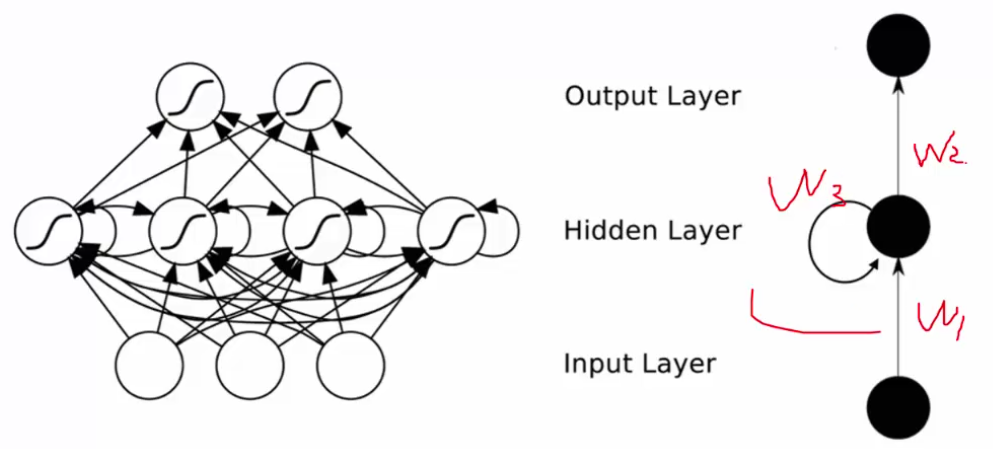
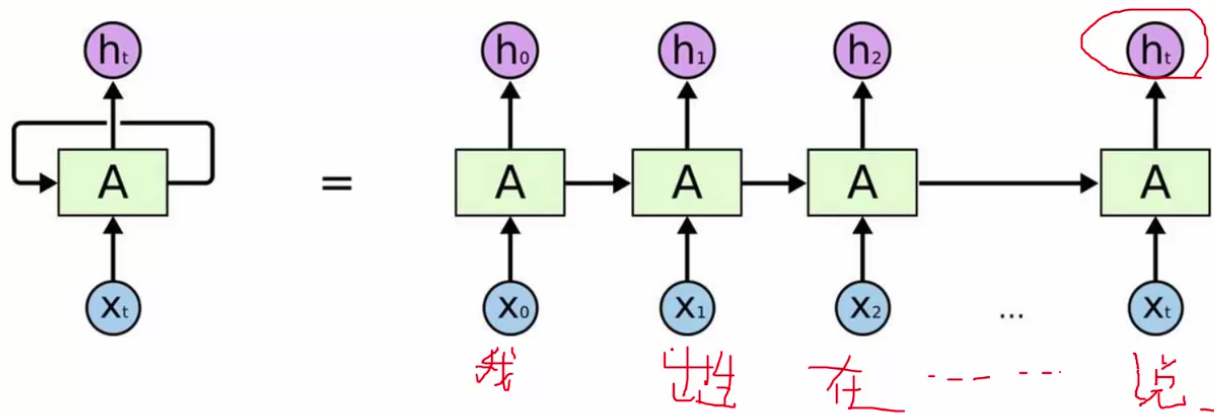
**011、RNN网络结构**

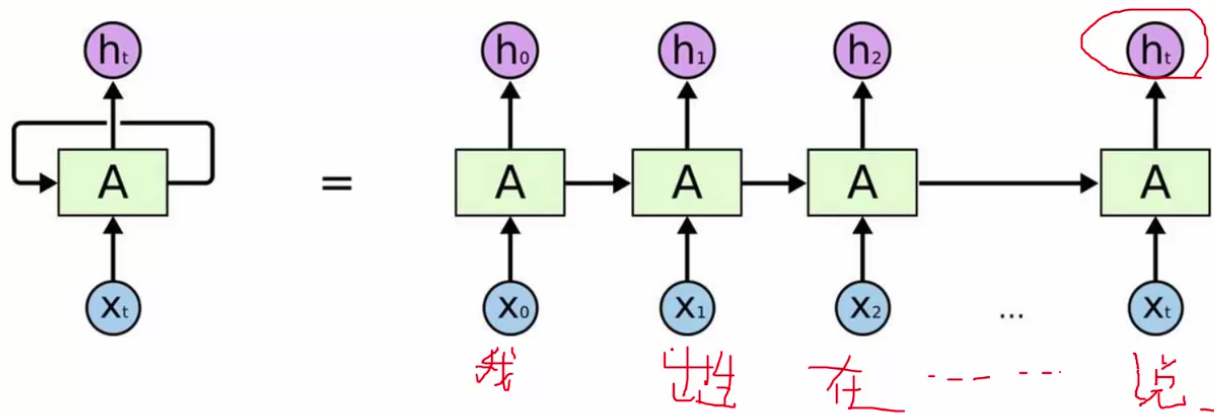


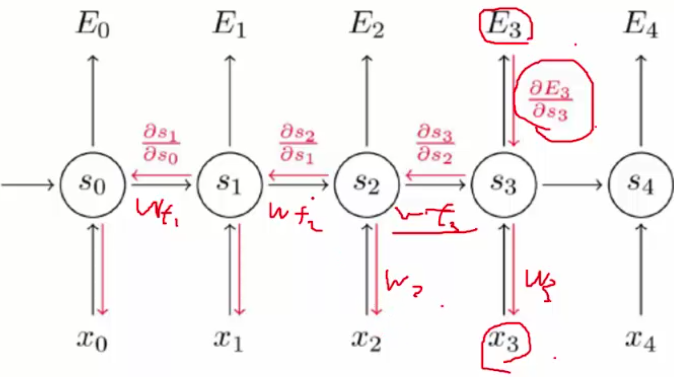




**012、RNN网络细节**

**适合NLP、ASR**





**RNN的问题：**

神经网络太庞大，计算量大；

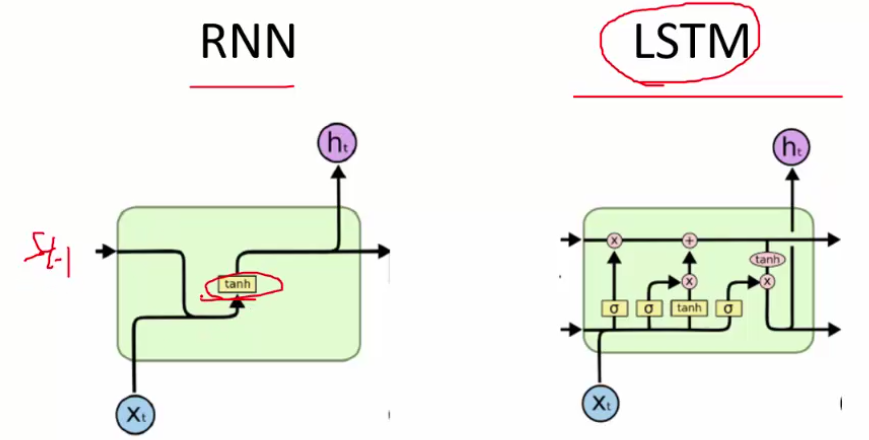
参数多容易造成梯度消失，如果某部分梯度很小，则相乘接近0。

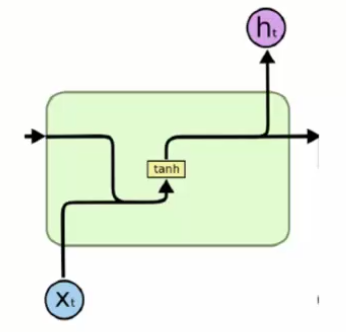
**013、Python实现RNN算法**

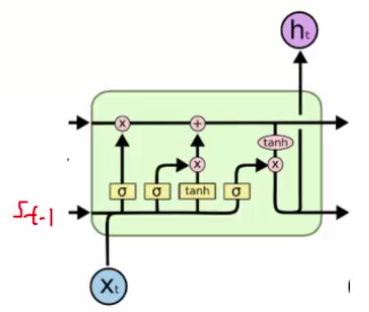
**014、LSTM网络结构简介**

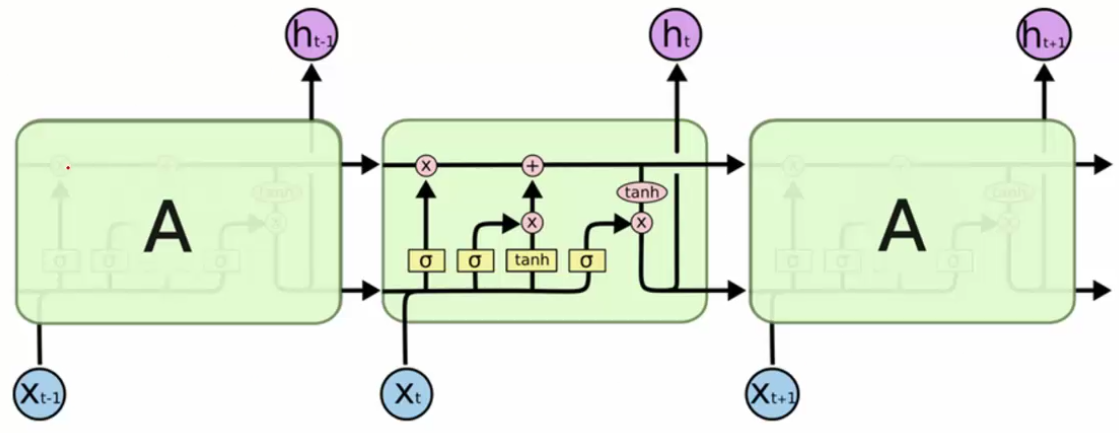
**LSTM是如何解决长时依赖问题的？（讲的清晰）**

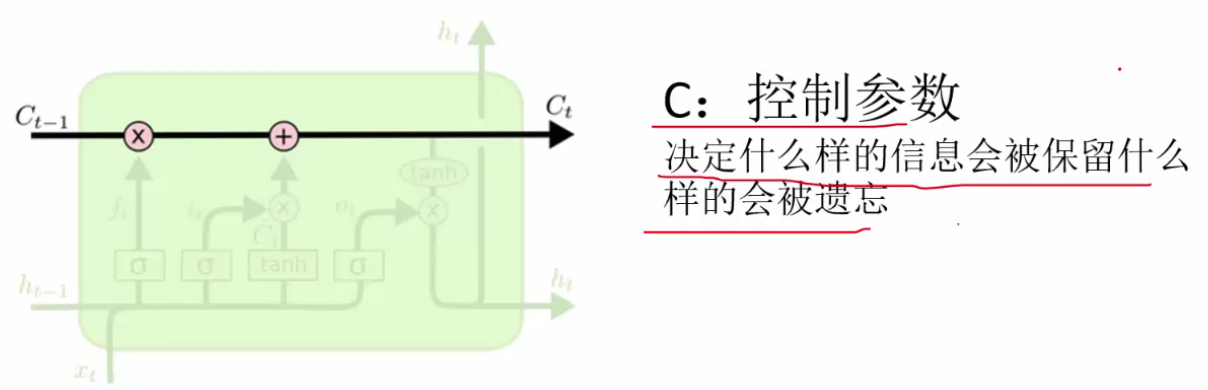
**https://www.jianshu.com/p/75eeaee7f67d**



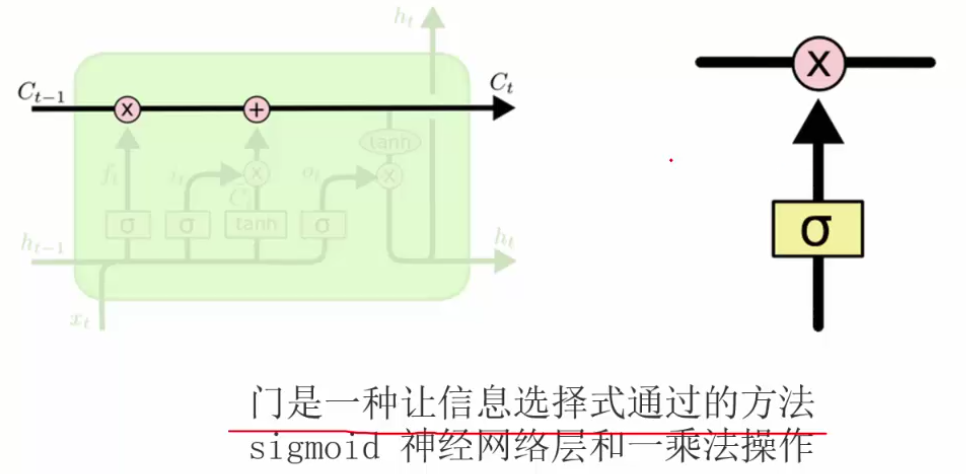




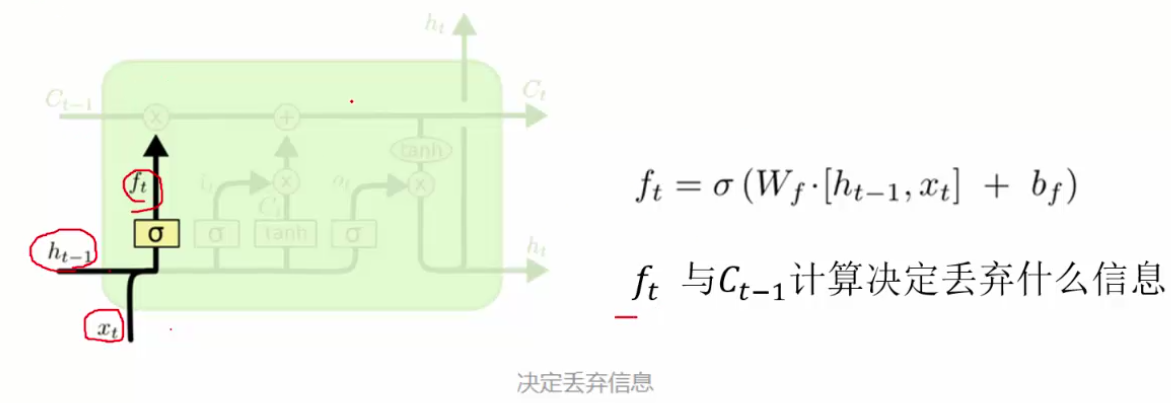




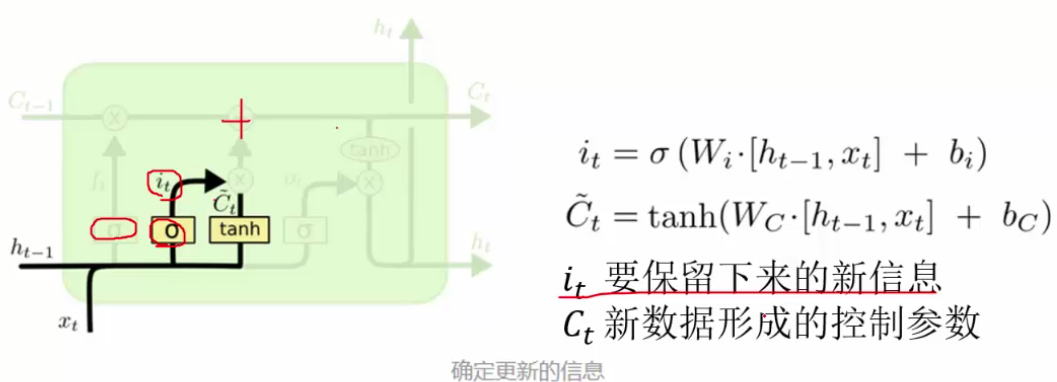
通过sigmoid函数选择，0或者1，遗忘或者记住



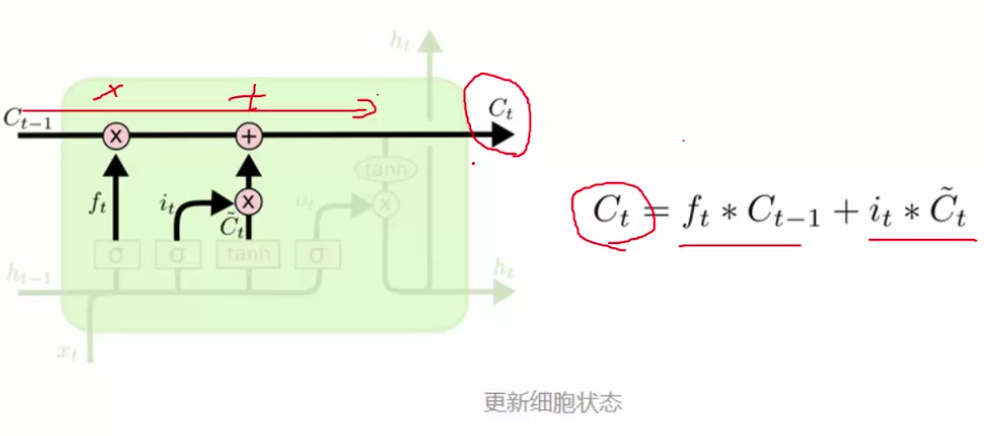
遗忘信息：



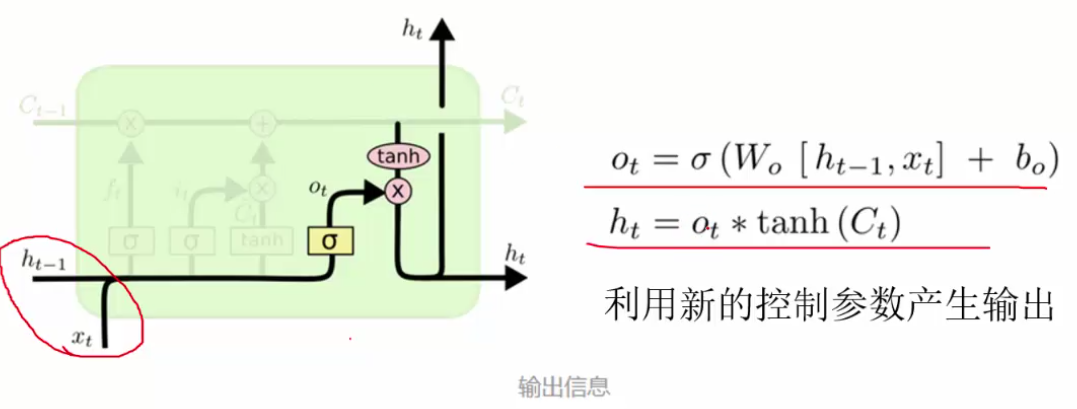
保留信息：

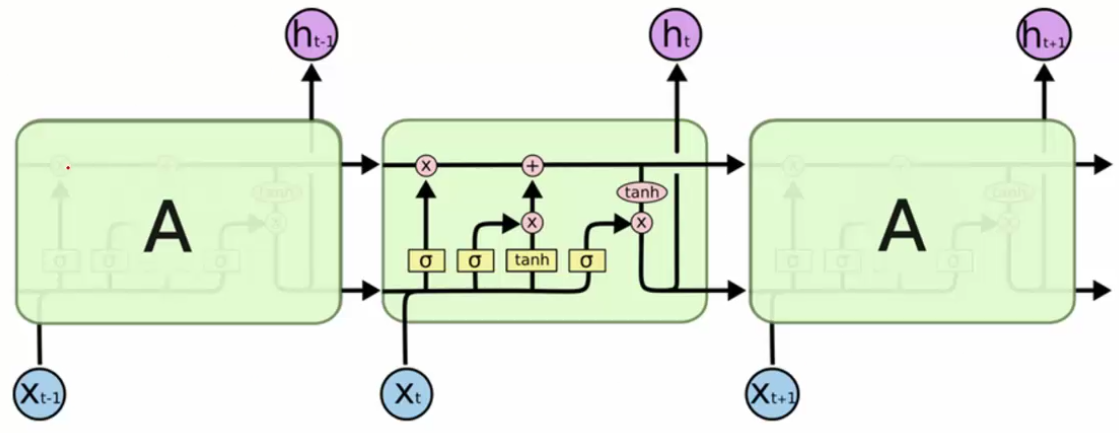


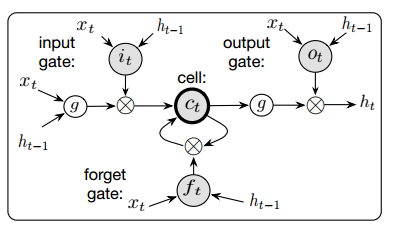
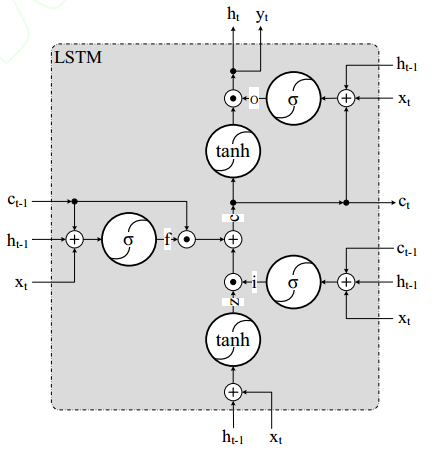
Ct一直都在更新：



输出：









其中，i、f、c、o分别代表**输入门、遗忘门、细胞状态、输出门**，输入门从Xt中读取输入，输出门将输出写入到ht，遗忘门可以重置单元变量，从而遗忘存储的输入信息Ct，ht-1是前一步的隐藏状态向量，W和b分别为对应权重系数矩阵和偏置项，和分别为和双曲正切激活函数，控制每个门打开或者关闭。其中，⊙表示按元素乘法（element-wise）的方式运算。每个内存块可以被视为一个独立

的单元。

LSTM 模型训练过程采用的是与经典的反向传播算法(BP)类似的BPTT算法，可分为四个步骤:

(1)按照前向计算方法(公式(3)-(7))计算 LSTM 细胞的输出值；

(2)反向计算每个 LSTM 细胞的误差项，包括按时间和网络层级两个反向传播方向；

(3)根据相应的误差项，计算每个权重的梯度；

(4)应用基于梯度的优化算法更新权重。