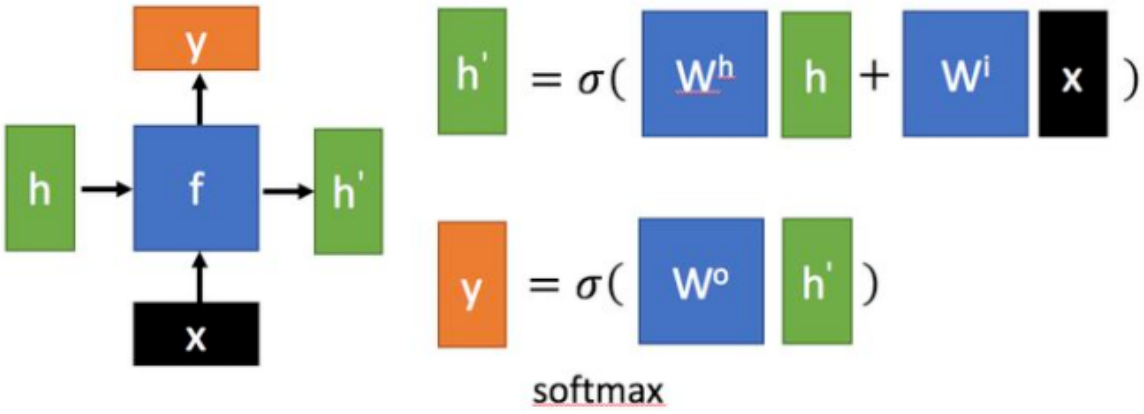
LSTM

RNN

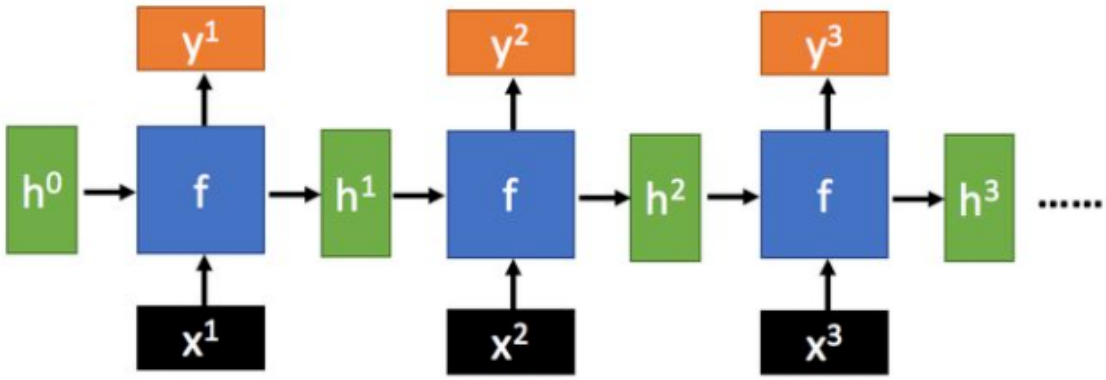
- 循环神经网络（RNN）是一种用于处理序列数据的神经网络，能够处理序列变化的数据。比如某个单词的意思会因为上下文提到的内容不同而有不同的含义，RNN能够很好的解决这类问题。



`x`为当前状态下数据的输入，`h`表示接收到的上一个节点的输入。

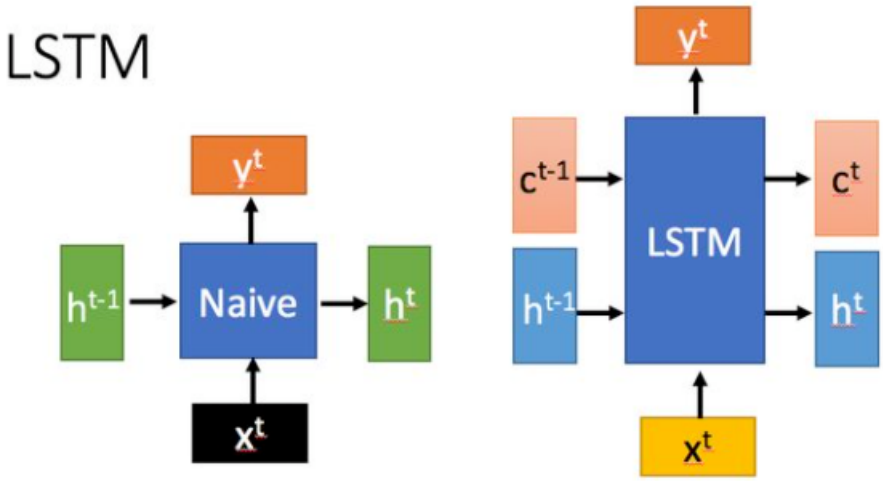
`y`为当前节点状态下的输出，`h'`为传递到下一个节点的输出。

下图是RNN结构图：



LSTM

- 长短期记忆（LSTM）是一种特殊的RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。相比普通的RNN，LSTM能够在更长的序列中有更好的表现。
- LSTM结构和普通的RNN的主要输入输出区别如下所示：

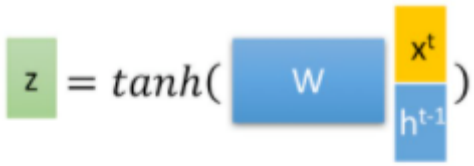


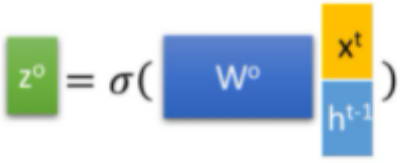
相比RNN只有一个传递状态 h^t ，LSTM有两个传输状态，一个 c^t和一个 h^t 。（RNN中的 h^t 对应于LSTM中的 c^t）

其中对于传递下去的 c^t 改变得很慢，通常输出的 c^t 是上一个状态传过来的 c^{t-1} 加上一些数值。而 h^t 则在不同节点下往往有很大的区别。

深入LSTM结构

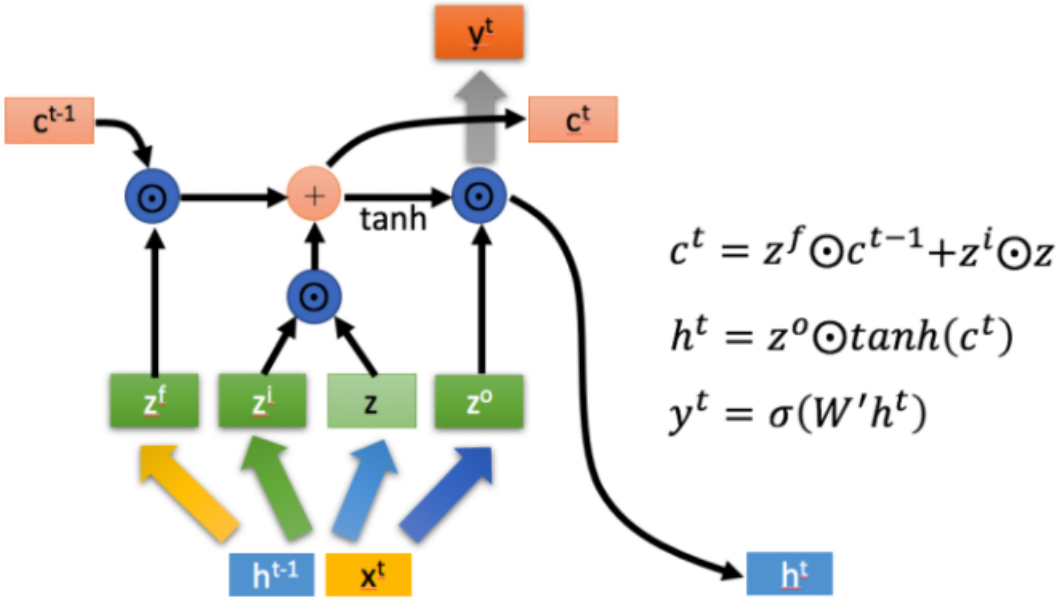
首先使用LSTM的当前输入 x^t 和上一个状态传递下来的 h^{t-1} 拼接训练得到四个状态。





其中， z^f ， z^i ， z^o 是由拼接向量乘以权值矩阵之后，再通过一个sigmoid激活函数转换成0到1之间的数值，来作为一种门控状态。而z则是将结果通过一个tanh激活函数转换成-1到1之间的值。

下面将介绍着四个状态在LSTM内部的使用。



⊙ 是Hadamard Product，也就是操作矩阵中对应的元素相乘，因此要求两个相乘矩阵是同型的，⊕则代表进行矩阵加法。

LSTM内部主要有三个阶段：

1. 忘记阶段。这个阶段主要是对上一个节点传来的输入进行选择性忘记。也就是"忘记不重要的，记住重要的"。

   具体来说就是通过计算得到的 z^f （f表示forget)来作为忘记门控，来控制上一个状态的 c^{t-1} 哪些需要留哪些需要忘记。

2. 选择记忆阶段。这个阶段的输入有选择性的记忆。主要是会对输入 x^t 进行选择性记忆，哪些重要则着重记录下来，哪些不重要，则少记一些。当前的输入内容由前面计算得到的z表示。而选择的门控信号则是由 z^i （i代表information)来进行控制。

   将上面两步得到的结果相加，即可得到传输给下一个状态的c^t。

3. 输出阶段。这个阶段将决定哪些将会被当成当前状态的输出。主要是通过z^o来进行控制的，并且还对上一阶段得到的`c`进行了放缩（tanh函数)。

与普通RNN类似，输出 y^t 往往最终也是通过 h^t 变化得到。

总结

通过门控状态来控制传输状态，记住需要长时间记忆的，忘记不重要的信息；而不像普通的RNN那样仅有一种记忆叠加方式。对很多需要长期记忆的任务来说，尤其好用。

但也因为引入了很多内容，导致参数变多，也使得训练难度加大。因此很多时候会使用效果和LSTM相当但参数更少的GRU来构建大训练量大的模型。