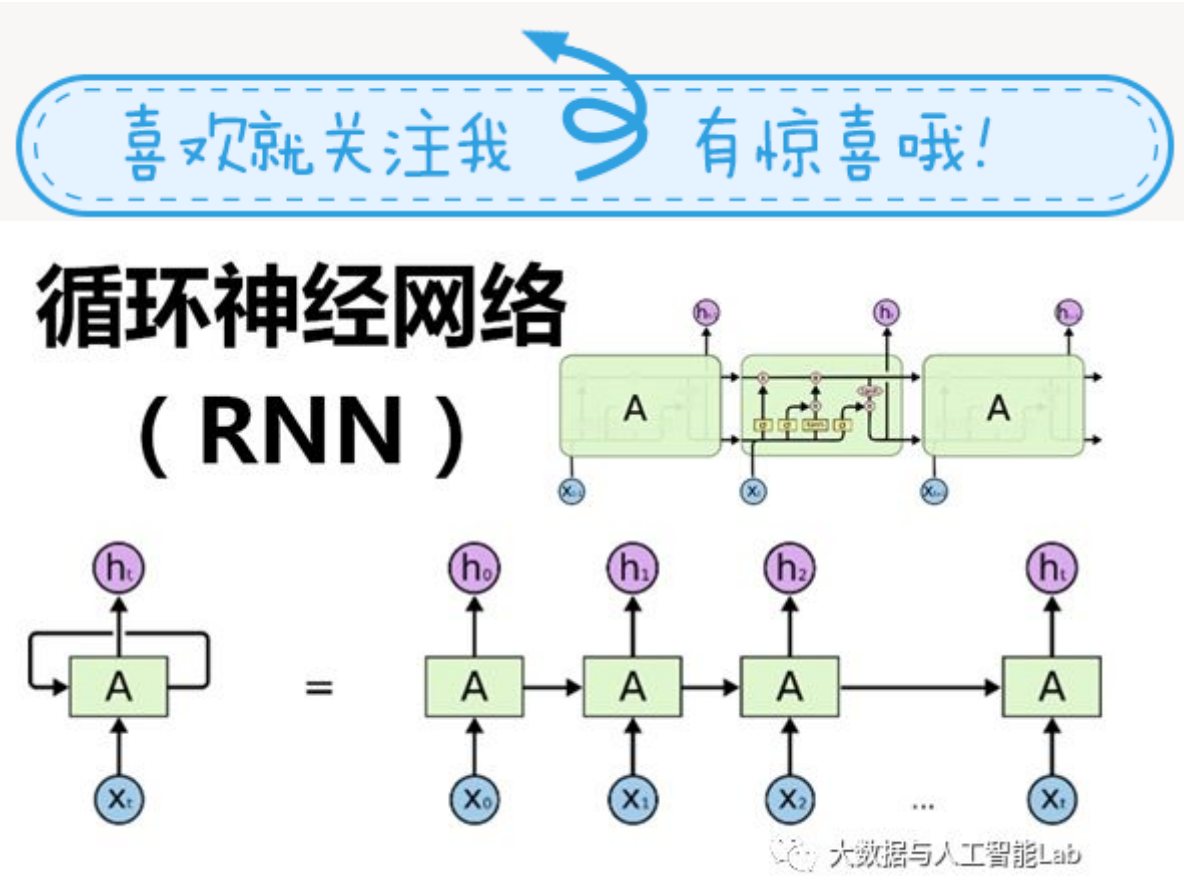
白话循环神经网络（RNN）

原创 雪饼 大数据与人工智能Lab 2018-02-13

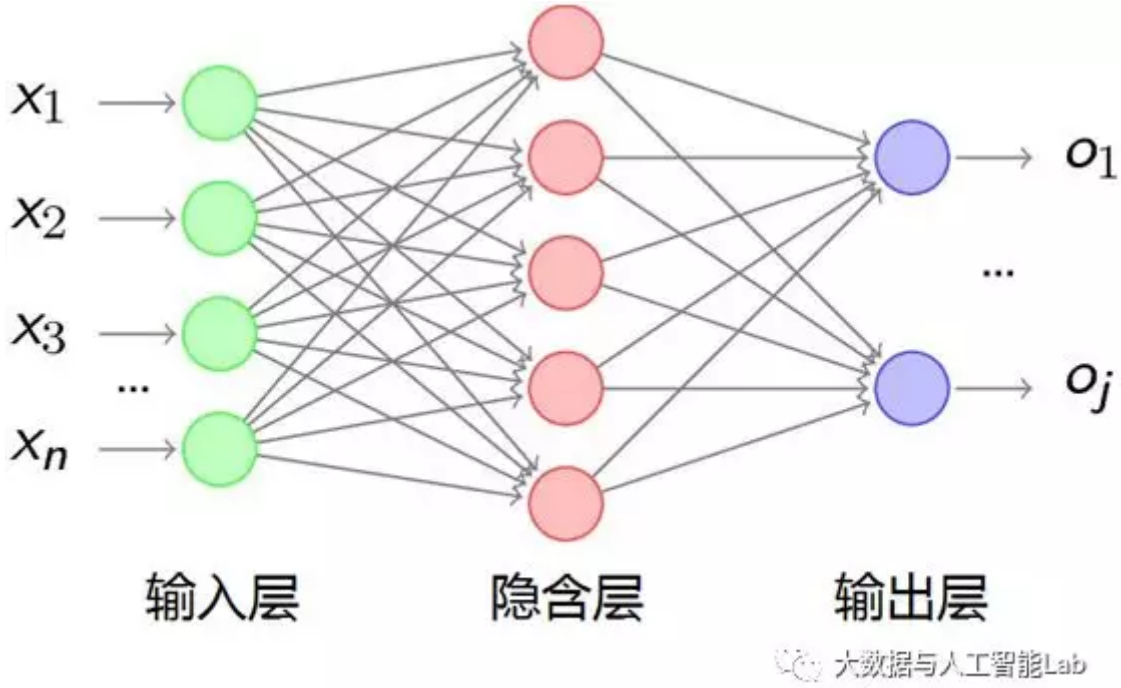


在上一篇文章中，介绍了 卷积神经网络（CNN）的算法原理，CNN在图像识别中有着强大、广泛的应用，但有一些场景用CNN却无法得到有效地解决，例如：

- **语音识别**，要按顺序处理每一帧的声音信息，有些结果需要根据上下文进行识别；
- **自然语言处理**，要依次读取各个单词，识别某段文字的语义

这些场景都有一个特点：就是都与时间序列有关，且输入的序列数据长度是不固定的。

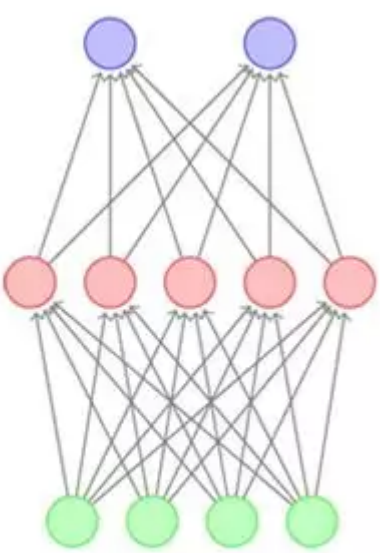
而经典的人工神经网络、深度神经网络（DNN），甚至卷积神经网络（CNN），一是输入的数据维度相同，另外是各个输入之间是独立的，每层神经元的信号只能向上一层传播，样本的处理在各个时刻独立。



而在现实生活中，例如对一个演讲进行语音识别，那演讲者每讲一句话的时间几乎都不太相同，而识别演讲者的讲话内容还必须要按照讲话的顺序进行识别。

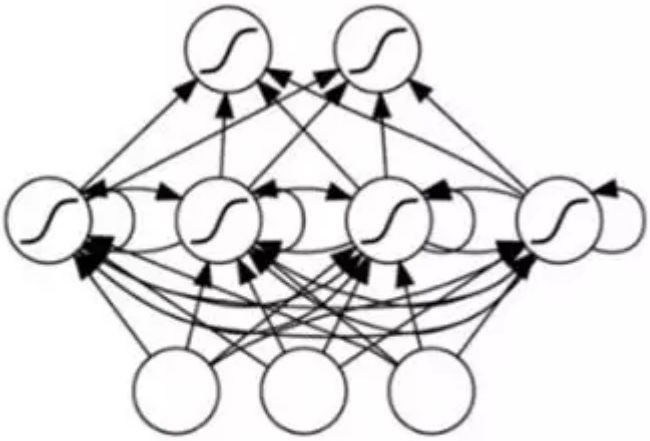
这就需要有一种能力更强的模型：该模型具有一定的记忆能力，能够按时序依次处理任意长度的信息。这个模型就是今天的主角：**“循环神经网络”（Recurrent Neural Networks，简称RNN）**。

循环神经网络（RNN），神经元的输出可以在下一个时间戳直接作用到自身（作为输入），看看下面的对比图：

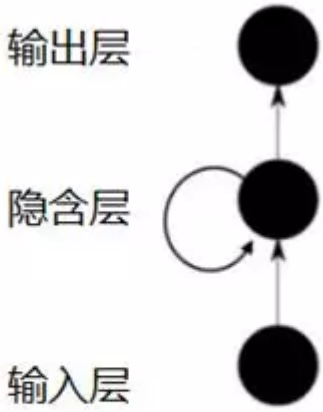


经典神经网络

简化图

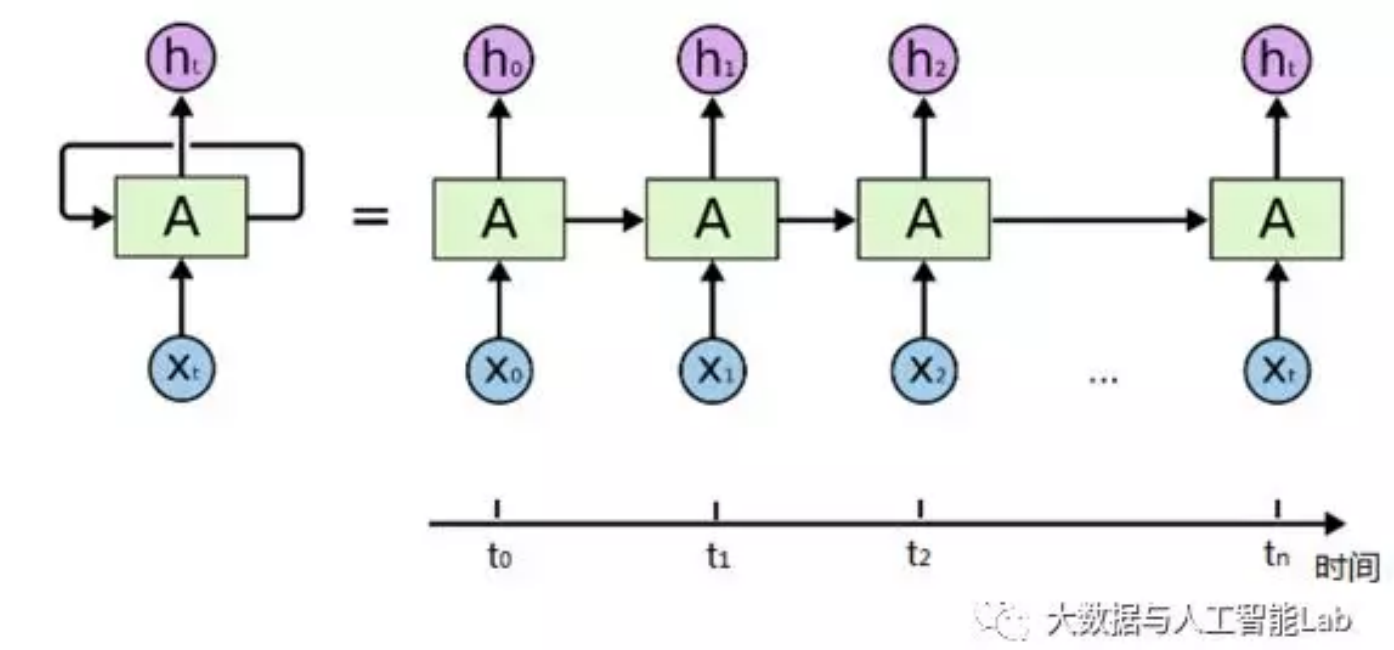


RNN 网络结构



RNN 简化图

从上面的两个简化图，可以看出RNN相比经典的神经网络结构多了一个循环圈，这个圈就代表着神经元的输出在下一个时间戳还会返回来作为输入的一部分，这些循环让RNN看起来似乎很神秘，然而，换个角度想想，也不比一个经典的神经网络难于理解。RNN可以被看做是对同一神经网络的多次赋值，第i层神经元在t时刻的输入，除了（i-1）层神经元在该时刻的输出外，还包括其自身在（t-1）时刻的输出，如果我们按时间点将RNN展开，将得到以下的结构图：

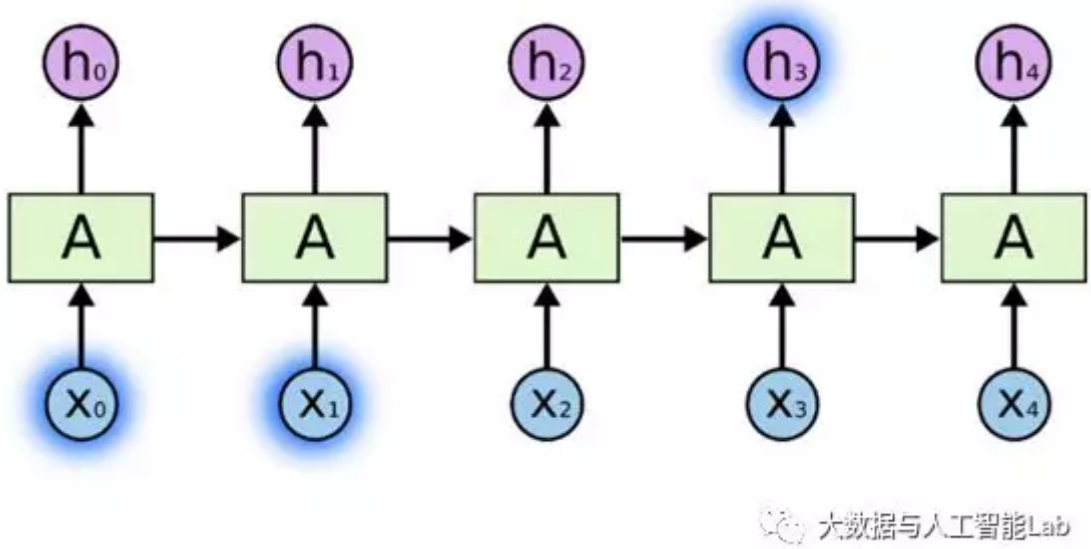


在不同的时间点，RNN的输入都与将之前的时间状态有关，tn时刻网络的输出结果是该时刻的输入和所有历史共同作用的结果，这就达到了对时间序列建模的目的。

【问题来了】关于RNN的长期依赖（Long-Term Dependencies）问题

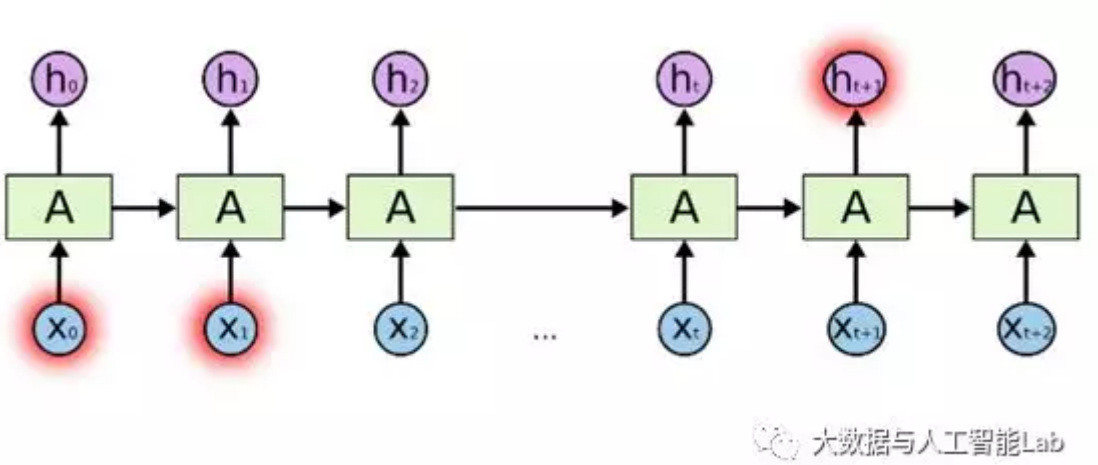
理论上，RNN可以使用先前所有时间点的信息作用到当前的任务上，也就是上面所说的长期依赖，如果RNN可以做到这点，将变得非常有用，例如在自动问答中，可以根据上下文实现更加智能化的问答。然而在现实应用中，会面临着不同的情况，例如：

（1）有一个语言模型是基于先前的词来预测下一个词。如果现在要预测以下这句话的最后一个单词"*白云飘浮在（天空）*"，我们并不需要任何其它上下文，最后一个词很显然就应该是"*天空*"。在这样的场景中，相关的信息和预测的词位置之间的间隔是非常小的，如下图所示：



（2）假设我们要预测"*我从小生长在四川.....我会讲流利的（四川话）*"最后一个词，根据最后一句话的信息建议最后一个词可能是一种语言的名字，但是如果我们要弄清楚是什

么语言，则需要找到离当前位置很远的“*四川*”那句话的上下文。这说明相关信息和当前预测位置之间的间隔就变得相当大。如下图所示：

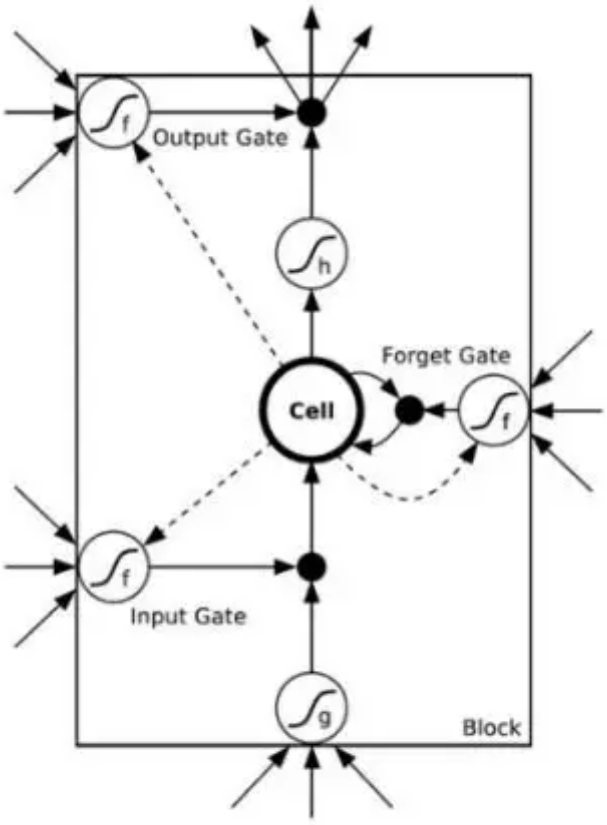


不幸的是，随着间隔的不断增大，RNN会出现“**梯度消失**”或“**梯度爆炸**”的现象，这就是RNN的长期依赖问题。例如我们常常使用sigmoid作为神经元的激励函数，如对于幅度为1的信号，每向后传递一层，梯度就衰减为原来的0.25，层数越多，到最后梯度指数衰减到底层基本上接受不到有效的信号，这种情况就是“梯度消失”。因此，随着间隔的增大，RNN会丧失学习到连接如此远的信息的能力。

【肿么办】神器来了：Long Short Term Memory网络（简称LSTM，长短期记忆网络）

LSTM是一种RNN特殊的类型，可以学习长期依赖信息。在很多问题上，LSTM都取得相当巨大的成功，并得到了广泛的应用。

一个LSTM单元的结构，如下图所示：



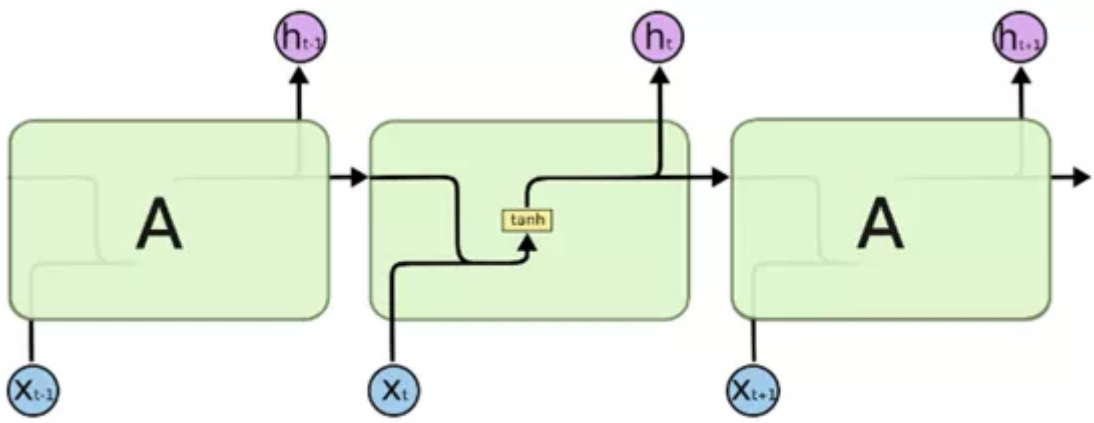
LSTM 单元

从上图可以看出，中间有一个cell（细胞），这也是LSTM用于判断信息是否有用的“处理器”。同时，cell旁边被放置了三扇门，分别是**输入门（Input Gate）、遗忘门（Forget Gate）和输出门（Output Gate）**。一个信息进入LSTM的网络当中，可以根据规则来判断是否有用，只有符合要求的信息才会被留下，不符合的信息则会通过遗忘门被遗忘。

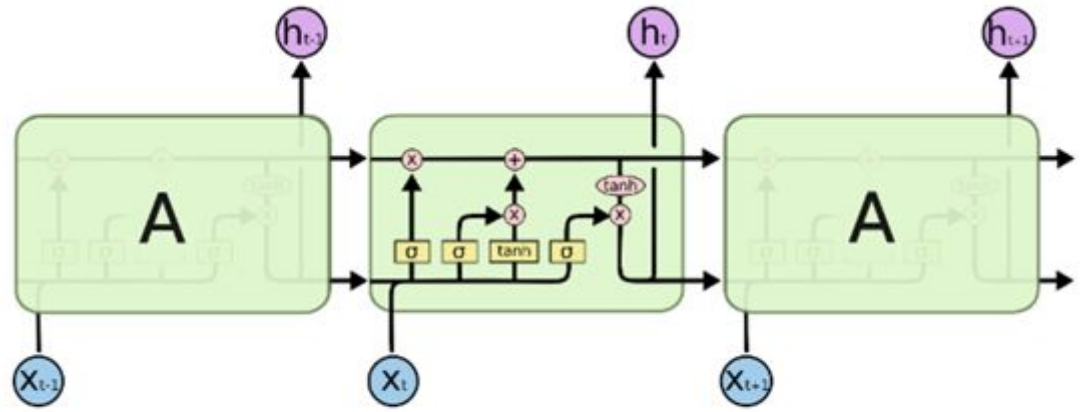
LSTM巧妙地通过“门”的形式，利用开关实现时间上的记忆功能，是解决长期依赖问题的有效技术。在数字电路中，门（gate）是一个二值变量{0,1}，0代表关闭状态、不允许任何信息通过；1代表开放状态，允许所有信息通过。而LSTM中的“门”也是类似，但它是一个“软”门，介于（0,1）之间，表示以一定的比例使信息通过。

一听起来就不明觉厉，那它是怎么做到的呢？

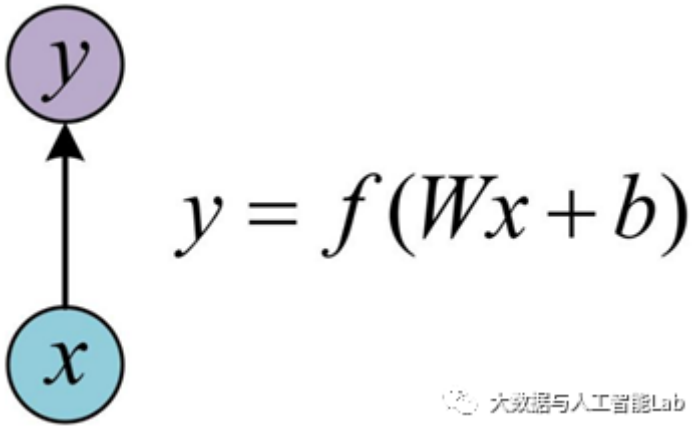
我们先来看一下RNN按时间展开后的简化图，结构很简单，标准RNN中的重复模块只包含单一的层，例如tanh层，如下图：



LSTM有着类似的结构，但是重复的模块拥有一个不同的结构，LSTM 中的重复模块包含四个交互的层，其中输入门（Input Gate）、遗忘门（Forget Gate）和输出门（Output Gate）便在这里面，如下图：



下面介绍一下LSTM的工作原理，下面会结合结构图和公式进行介绍，回顾一下最基本的单层神经网络的结构图、计算公式如下，表示输入是x，经过变换Wx+b和激活函数f得到输出y。下面会多次出现类似的公式

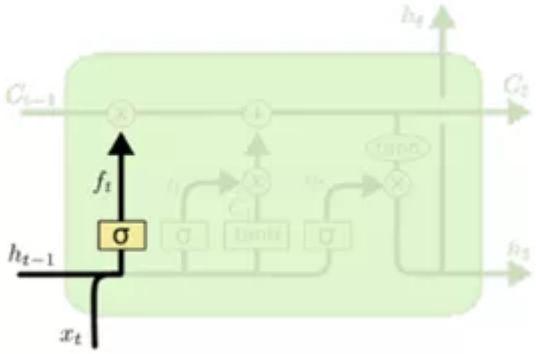


下面以一个语言模型的例子来进行介绍，这个模型是根据已经看到的词来预测下一个词，例如：

小明刚吃完米饭，现在准备要吃水果，然后拿起了一个（　　）

（1）遗忘门（Forget Gate）

该门的示意图如下，该门会读取ht-1和xt的信息，通过sigmoid层输出一个介于0 到 1 之间的数值，作为给每个在细胞状态Ct-1中的数字，0 表示“完全舍弃”，1 表示“完全保留”。



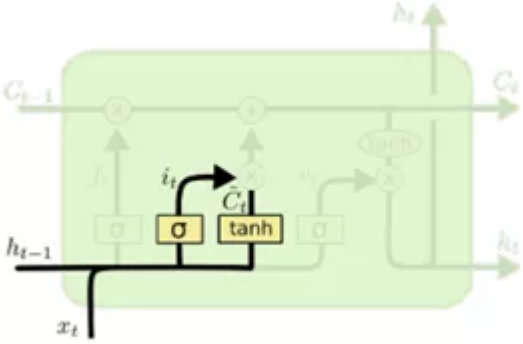
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

结合上面讲到的语言预测模型例子，*“小明刚吃完米饭”*，这句话主语是*“小明”*，宾语是*“米饭”*，下一句话*“现在准备要吃水果”*，这时宾语已经变成了新的词*“水果”*，那第三句话要预测的词，就是跟*“水果”*有关了，跟*“米饭”*已经没有什么关系，因此，这时便可以利用“遗忘门”将*“米饭”*遗忘掉。

（2）输入门（Input Gate）

下一步是确定什么样的新信息被存放在细胞状态中。这里包含两部分：

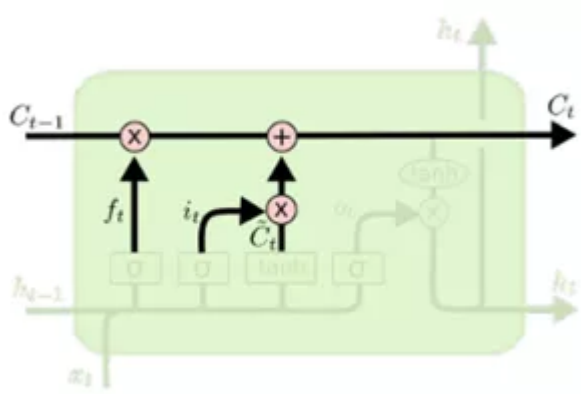
首先是经过“输入门”，这一层是决定我们将要更新什么值；然后，一个 tanh 层创建一个新的候选值向量，加入到状态中，如下图：



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

在这个语言预测模型的例子中，我们希望将新的代词*“水果”*增加到细胞状态中，来替代旧的需要忘记的代词*“米饭”*。

现在来更新旧细胞的状态，由Ct-1更新为Ct，更新方式为：（1）把旧状态Ct-1与ft相乘（回顾一下，ft就是遗忘门，输出遗忘程度，即0到1之间的值），丢弃掉需要丢弃的信息（如遗忘门输出0，则相乘后变成0，该信息就被丢弃了）；（2）然后再加上it与候选值相乘（计算公式见上图）。这两者合并后就变成一个新的候选值。

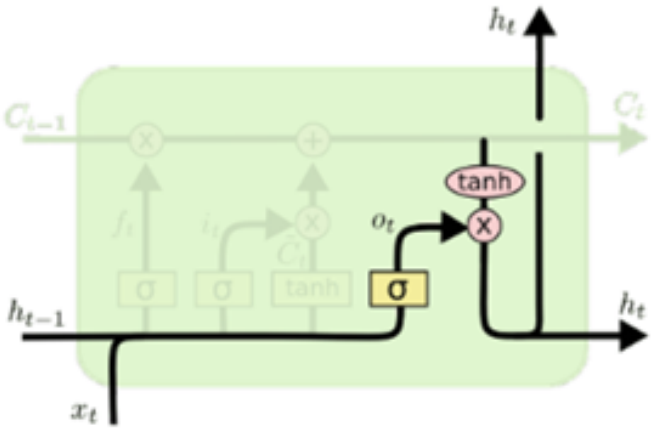


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

在这个语言预测模型的例子中，这就是根据前面确定的目标，丢弃旧的代词信息（*米饭*）并添加新的信息（*水果*）的地方。

（3）输出门（Output Gate）

最后我们要确定输出什么值，首先，通过一个sigmoid层来确定细胞状态的哪个部分将要输出出去，接着，把细胞状态通过 tanh 进行处理（得到一个介于-1到1之间的值）并将它和 sigmoid的输出结果相乘，最终将会仅仅输出我们需要的那部分信息。

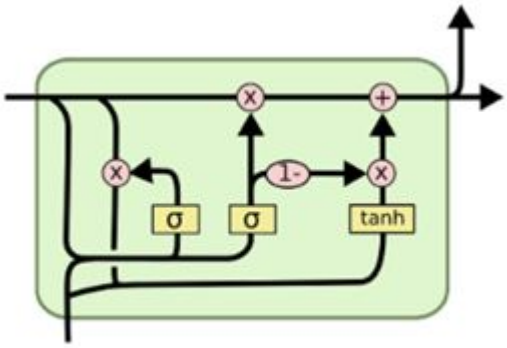


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

在这个语言模型的例子中，因为看到了一个新的代词（*水果*），可能需要输出与之相关的信息（*苹果、梨、香蕉……*）。

以上就是标准LSTM的原理介绍，LSTM也出现了不少的变体，其中一个很流行的变体是**Gated Recurrent Unit (GRU)**，它将遗忘门和输入门合成了一个单一的更新门，同样还混合了细胞状态和隐藏状态，以及其它一些改动。最终GRU模型比标准的 LSTM 模型更简单一些，如下图所示：



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

大数据与人工智能Lab

相关阅读

- 白话卷积神经网络（CNN）
- 吴恩达《机器学习》课程
- AlphaGo算法原理浅析
- 浅说"迁移学习"（Transfer Learning）
- 什么是强化学习（Reinforcement Learning）

搜索"大数据与人工智能Lab"微信号（BigdataAILab），或扫描二维码关注我们

你的每一份支持将是我们最大的动力，感谢你的支持，感谢你的赞赏