深度学习笔记 | 第11讲：写一篇人人都能看得懂的LSTM

louwill 狗熊会 2018-10-29

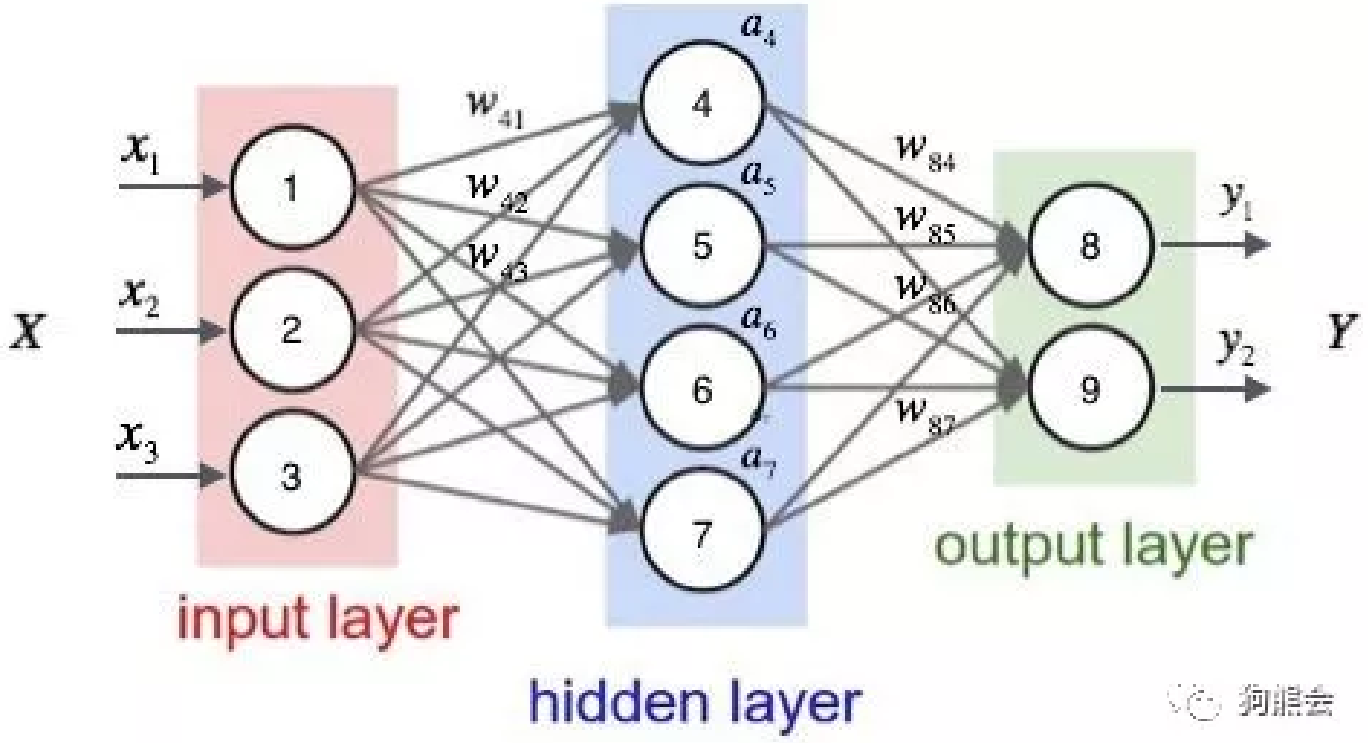
大家好！又到了每周一狗熊会的深度学习时间了。从上一讲开始，小编和大家一起将目光由 CNN 转向了神经网络的另一大类型循环神经网络 RNN，对 RNN 网络的基本构造、原理以及应用领域进行了详细的介绍。在本讲中，小编将继续和大家介绍 RNN 的一种著名的变种网络——LSTM（长短期记忆网络）。小编力图用详细但又不失明了的语言来给大家讲懂复杂的 LSTM 网络。

——1——

深度神经网络的困扰：梯度爆炸与梯度消失

在此前的普通深度神经网络和深度卷积网络的讲解时，当网络结构变深时，神经网络在训练时碰到梯度爆炸或者梯度消失的情况。那么什么是梯度爆炸和梯度消失呢？它们又是怎样产生的呢？问题先抛出，咱们

一个一个来看。



大家都了解了神经网络的训练机制，不管是哪种类型的神经网络，其训练都是通过反向传播计算梯度来实现权重更新的。我们通过设定损失函数，建立损失函数关于各层网络输入输出的梯度计算，当网络训练开动起来的时候，系统便按照反向传播机制来不断更新网络各层参数直到停止训练。但当网络层数加深时，这个训练系统并不是很稳，经常会出现一些幺蛾子。其中梯度爆炸和梯度消失便是最大的幺蛾子之二。

所谓梯度爆炸就是在神经网络训练过程中，梯度变得越来越大以使得神经网络权重得到疯狂更新的情形，这种情况很容易发现，因为梯度过大，计算更新得到的参数也会大到崩溃，这时候我们可能看到更新的参数值中有很多的 NaN，这说明梯度爆炸已经使得参数更新出现数值溢出。这便是梯度爆炸的基本情况。

再来看梯度消失。与梯度爆炸相反的是，梯度消失就是在神经网络训练过程中梯度变得越来越小以至于梯度得不到更新的一种情形。当网络加深时，网络深处的误差很难因为梯度的减小很难影响到前层网络的权重更新，一旦权重得不到有效的更新计算，神经网络的训练机制也就失效了。

那么大家可能又要问了，神经网络训练过程中梯度怎么就会变得越来越大或者越来越小呢？咱们以本系列的第一讲的神经网络反向传播推导公式为例来解释：

$$\frac{\partial L}{\partial a_2} = \frac{d}{da_2} L(a_2, y) = \left(-y log a_2 - (1-y) log(1-a_2)\right)' = -\frac{y}{a_2} + \frac{1-y}{1-a_2}$$

$$\frac{\partial L}{\partial Z_2} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} = a_2 - y$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} \frac{\partial Z_2}{\partial W_2} = \frac{1}{m} \frac{dL}{dZ_2} a_1 = \frac{1}{m}(a_2 - y) a_1$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} \frac{\partial Z_2}{\partial b_2} = \frac{dL}{dZ_2} = a_2 - y$$

$$\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} \frac{\partial Z_2}{\partial a_1} = (a_2 - y) w_2$$

$$\frac{\partial L}{\partial Z_1} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} \frac{\partial Z_2}{\partial a_1} \frac{\partial a_1}{\partial Z_1} = (a_2 - y) w_2 \sigma'(Z_1)$$

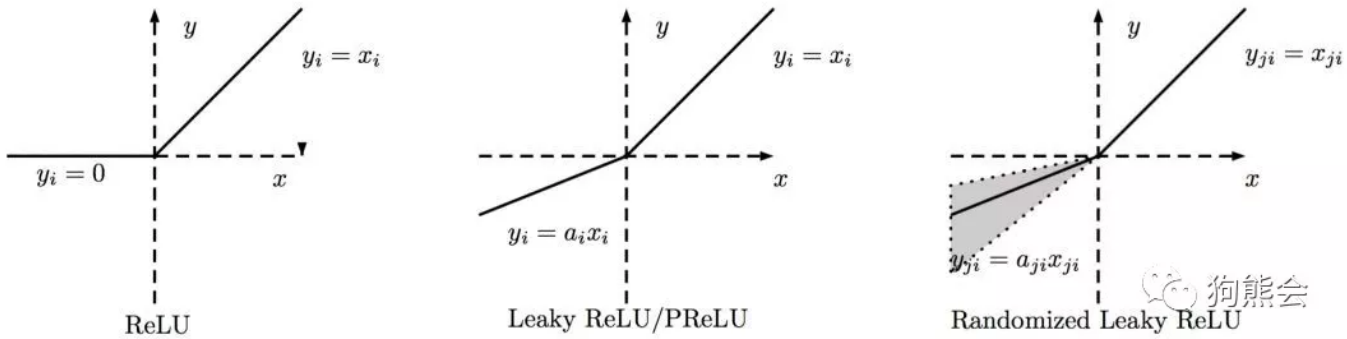
$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} \frac{\partial Z_2}{\partial a_1} \frac{\partial a_1}{\partial Z_1} \frac{\partial Z_1}{\partial w_1} = (a_2 - y) w_2 \sigma'(Z_1) x$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial Z_2} \frac{\partial Z_2}{\partial a_1} \frac{\partial a_1}{\partial Z_1} \frac{\partial Z_1}{\partial b_1} = (a_2 - y) w_2 \sigma'(Z_1)$$

上述公式是一个两层网络的反向传播参数更新公式推导过程。离输出层相对较远的是输入到隐藏层的权重参数，可以看到损失函数对于隐藏层输出 a1输入到隐藏层权重W1和偏置b1的梯度计算公式，一般而言都会转换从下一层的权重乘以激活函数求导后的式子。如果激活函数求导后的结果和下一层权重的乘积大于1或者说远远大于1的话，在网络层数加深时，层层递增的网络在做梯度更新时往往就会出现梯度爆炸的情况。如果激活函数求导和下一层权重的乘积小于1的话，在网络加深时，浅层的网络梯度计算结果会越来越小往往就会出现梯度消失的情况。所以可是说是反向传播的机制本身造就梯度爆炸和梯度消失这两种不稳定因素。比如说一个 100 层的深度神经网络，假设每一层的梯度计算值都为1.1，经过由输出到输入的反向传播梯度计算可能最后的梯度值就变成 1.1^100 = 13780.61234，这是一个极大的梯度值了，足以造成计算溢出问题。若是每一层的梯度计算值为 0.9，反向传播输入层的梯度计算值则可能为 0.9^100 = 0.000026561398，足够小到造成梯度消失。（例子只是一个简化的假设情况，实际反向传播计算要更为复杂。）

所以总体来说，神经网络的训练中梯度过大或者过小引起的参数过大过小都会导致神经网络失效，那我们的目的就是要让梯度计算回归到正常的区间范围，不要过大也不要过小，这也是解决这两个问题的一个思路。下图是一个 4 层网络的训练过程各参数的取值范围：

那么梯度爆炸和梯度消失怎么解决呢？梯度爆炸并不麻烦，在实际训练的时候对梯度进行修剪即可，但是梯度消失的处理就比较麻烦了，由上述的分析我们知道梯度消失一个关键在于激活函数。sigmoid 激活函数本身就更容易产生这种幺蛾子，所以一般而言，我们换上更加鲁棒的 ReLu 激活函数以及给神经网络加上归一化激活函数层，一般问题都能得到很好的解决，但也不是任何情形下都管用，比如说咱们的 RNN 网络，具体在下文中我们在做集中探讨。

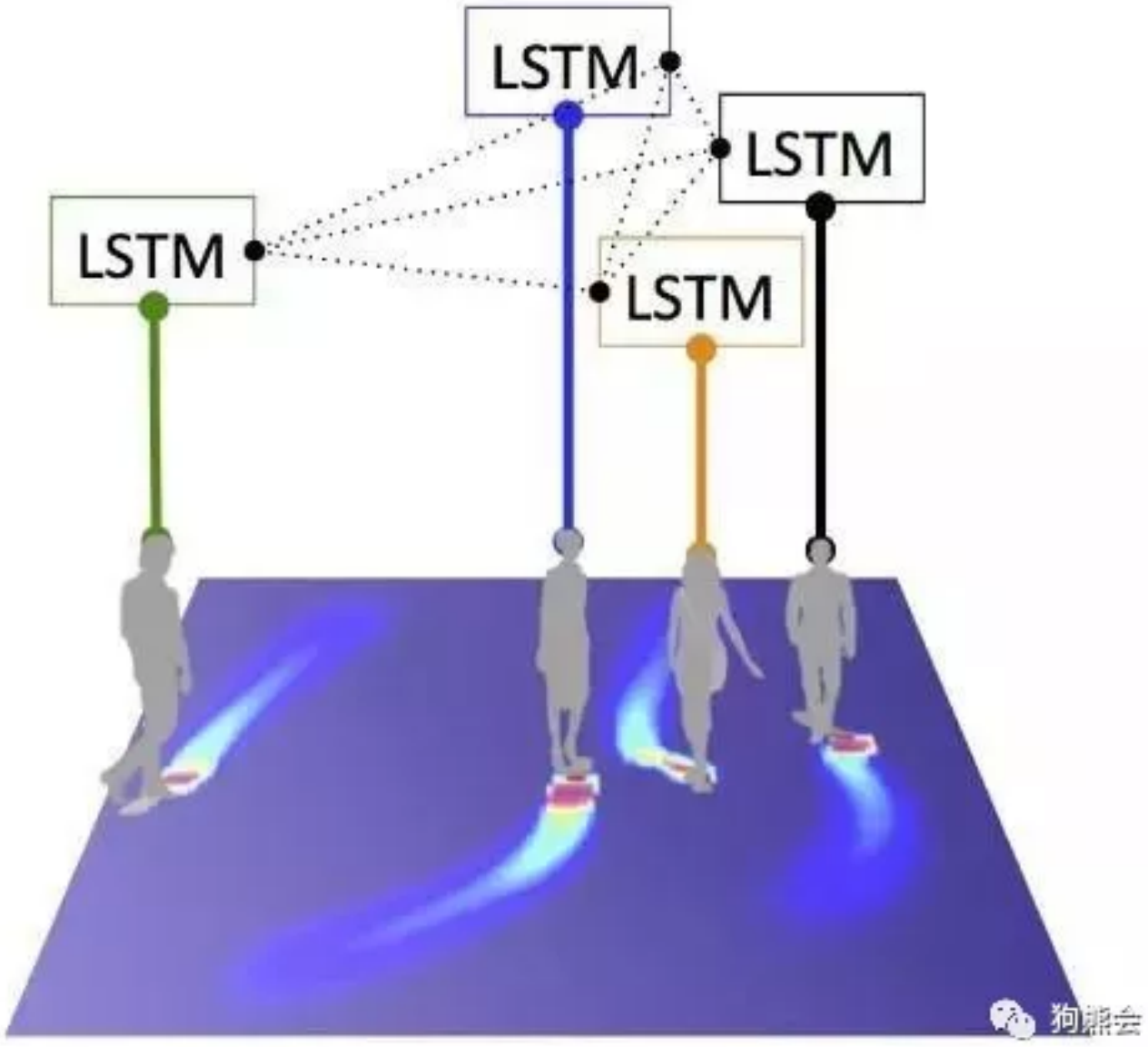


以上便是梯度爆炸和梯度消失这两种问题的基本解释，下面我们回归正题，来谈谈本文的主角——LSTM。

——2——

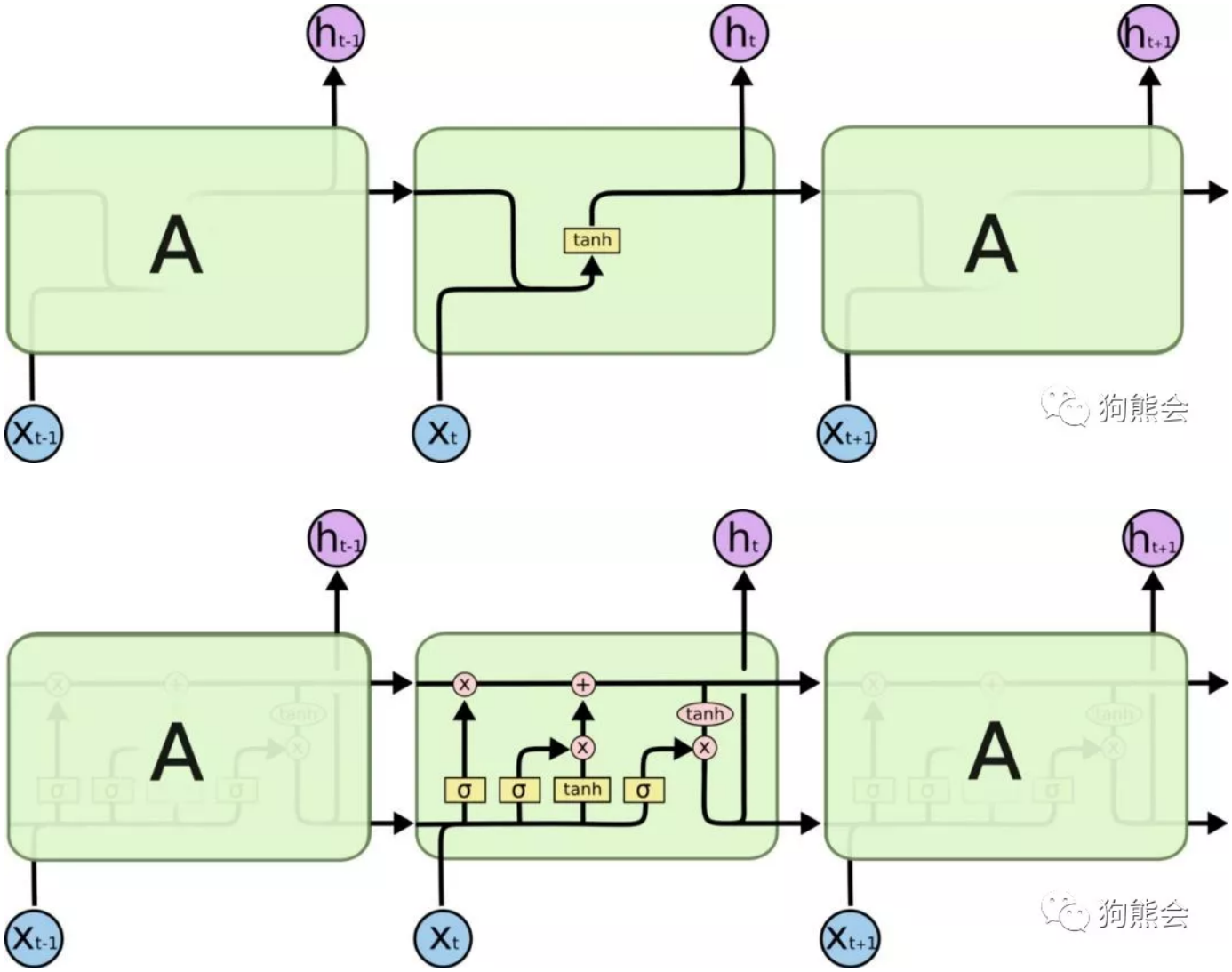
LSTM：让RNN具备更好的记忆机制

前面说了很多铺垫，全部都是为了来说本节的 LSTM。梯度爆炸和梯度消失，普通神经网络和卷积神经网络有，那么循环神经网络 RNN 也有吗？必须有。而且梯度消失和梯度爆炸的问题之于 RNN 来说伤害更大。当 RNN 网络加深时，因为梯度消失的问题使得前层的网络权重得不到更新，RNN 的记忆性就很难生效。为此，在传统的 RNN 网络结构基础上，研究人员给出一些著名的改进方案，因为这些改进方案都脱离不了经典的 RNN 架构，所以一般来说我们也称这些改进方案为 RNN 变种网络。比较著名的就是 GRU（循环门控单元）和 LSTM（长短期记忆网络）。GRU 和 LSTM 二者结构基本一致，但有些许不同，本节小编挑名气更大更有代表性的 LSTM 来进行讲述。

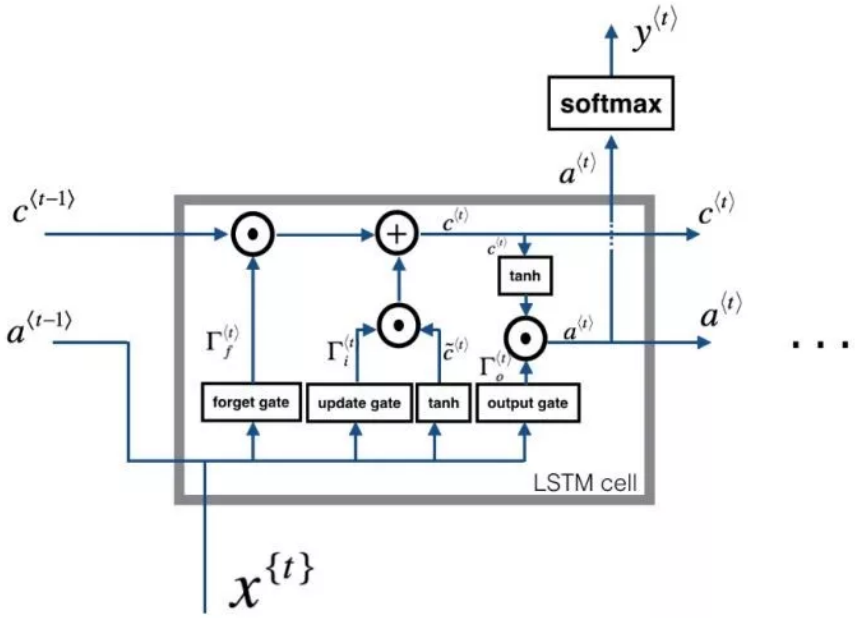


在正式讲 LSTM 的技术细节之前，小编先来给大家明确几点。第一，LSTM 的本质是一种 RNN 网络。第二，LSTM 在传统的 RNN 结构上做了相对复杂的改进，这些改进使得 LSTM 相对于经典 RNN 能够很好的解决梯度爆炸和梯度消失问题，让循环神经网络具备更强更好的记忆性能，这也是 LSTM 的价值所在。那咱们就来重点看一下 LSTM 的技术细节。

咱们先摆一张经典 RNN 结构与 LSTM 结构对比图，好让大家能够有一个宏观的了解，然后小编再针对 LSTM 结构图中各个部分进行拆解分析。下图是标准 RNN 结构与 LSTM 单元的结构对比：



可以看到，每个 LSTM 单元中包含了 4 个交互的网络层，完整的 LSTM 公式表示如下所示：

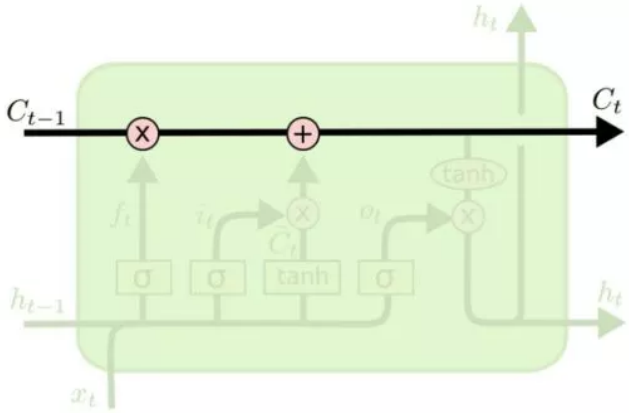


$$\Gamma_f^{\langle t\rangle} = \sigma(W_f[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_f)$$
$$\Gamma_u^{\langle t\rangle} = \sigma(W_u[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_u)$$
$$\tilde{c}^{\langle t\rangle} = \tanh(W_C[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_C)$$
$$c^{\langle t\rangle} = \Gamma_f^{\langle t\rangle} \circ c^{\langle t-1\rangle} + \Gamma_u^{\langle t\rangle} \circ \tilde{c}^{\langle t\rangle}$$
$$\Gamma_o^{\langle t\rangle} = \sigma(W_o[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_o)$$
$$a^{\langle t\rangle} = \Gamma_o^{\langle t\rangle} \circ \tanh(c^{\langle t\rangle})$$

下面我们根据结构图和公式来逐模块解释 LSTM。

- 记忆细胞 ct−1 -> ct

从图中可以看到在 LSTM 单元的最上层有一条贯穿的关于记忆细胞 ct−1 到 ct 的箭头直线。这样贯穿的直线表现记忆信息在网络各层之间保持下去很容易。



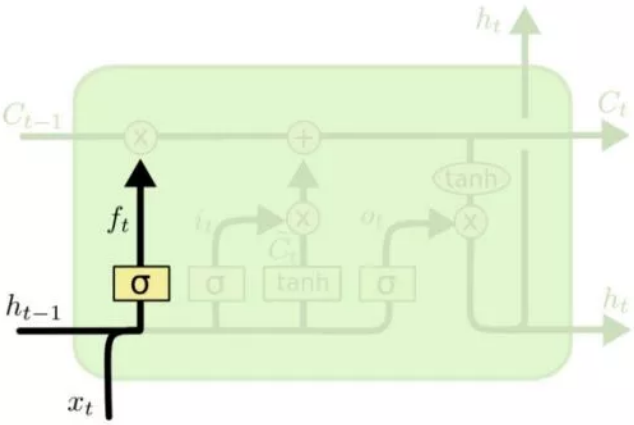
然后来看 LSTM 的第一个门控：遗忘门。

- 遗忘门（forget gate）

遗忘门的计算公式如下：

$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$

所谓遗忘门就是我们要决定从记忆细胞中是否丢弃某些信息，这个过程就是遗忘门要干的事，我们通过一个 sigmoid 函数来进行处理。遗忘门在整个结构中的位置如下图所示：



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

可以看到，遗忘门接受来自输入 xt 和上一层隐状态 ht−1 的值进行加权计算处理。

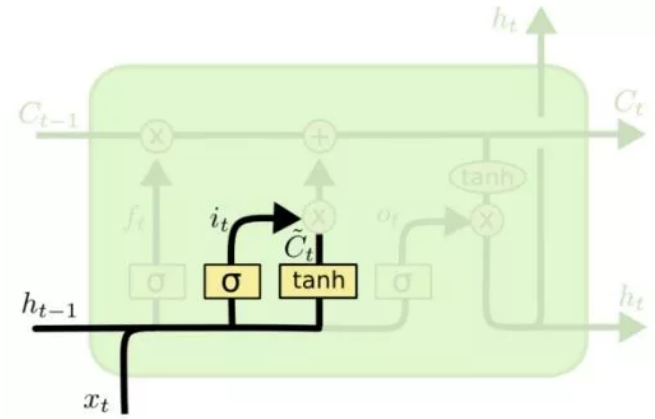
- 更新门（update gate）

然后就是更新门，我们需要确定什么样的信息能存入细胞状态中。这跟我们在 GRU 中类似，除了计算更新门之外，还需要通过 tanh 计算记忆细胞的候选值 ct_hat。 LSTM 中更新门需要更加细心一点。候选值和更新门的计算公式如下：

$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

$$\tilde{c}^{\langle t \rangle} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$

更新门在整个结构中的位置如下图所示：



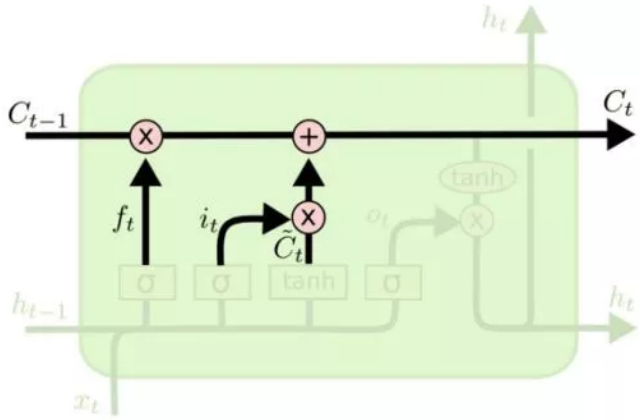
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

然后，LSTM 结合遗忘门、更新门、上一层记忆细胞值和记忆细胞候选值来共同决定和更新当前细胞状态：

$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$

在整个结构中位置如下图所示：



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

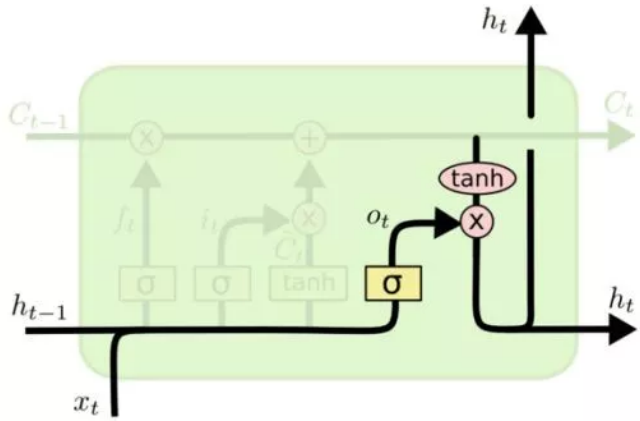
- 输出门（output gate）

LSTM 提供了单独的输出门。计算公式如下：

$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$

$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

输出门的位置如下图所示：



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

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

这便是完整的 LSTM 结构。虽然复杂，但经我们逐步解析之后也就基本清晰了。LSTM 在自然语言处理、问答系统、股票预测等等领域都有着广泛而又深入的应用，后面小编也将有选择性的给大家做一些分享。

以上便是本节的内容。

在本节内容中，小编和大家重点介绍了深度神经网络中普遍存在的梯度爆炸和梯度消失问题，对其形成原因和机制以及解决方法都进行了详细的讲述。并以此为基础对 LSTM 的网络架构和技术原理进行了深入讲解。在下一讲中，小编将继续分享关于 RNN 和 LSTM 在自然语言处理方面的应用案例。

【参考资料】

deeplearningai.com

https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

https://blog.csdn.net/qq_28743951/article/details/78974058

作者简介

鲁伟，狗熊会人才计划一期学员。目前在杭州某软件公司从事数据分析和深度学习相关的研究工作，研究方向为贝叶斯统计、计算机视觉和迁移学习。

识别二维码，查看作者更多精彩文章

点击“**阅读原文**”，成为狗熊会会员！

阅读原文