LSTM 长短时记忆网络 与 GRU

2021年2月9日　20:59

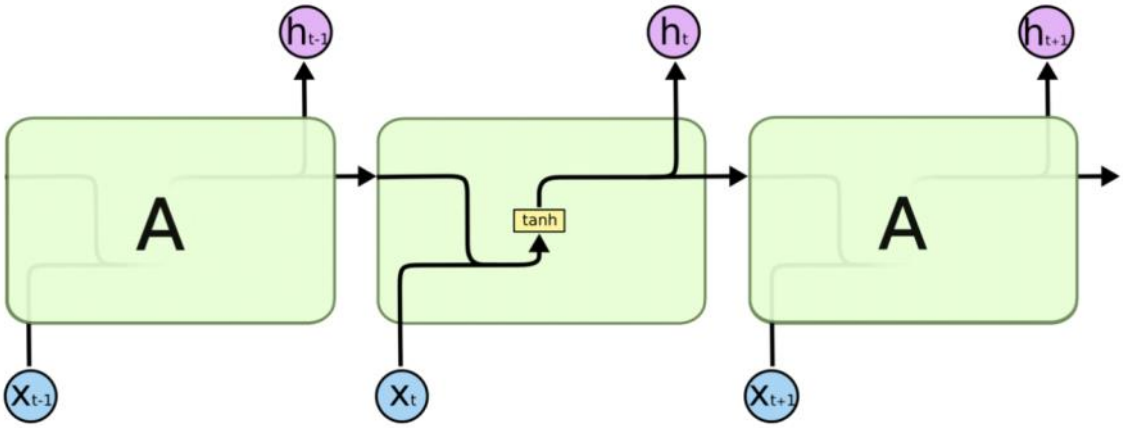
序

前面说过了，因为梯度消失的问题 RNN 的长时记忆功能十分有限。而 LSTM 长短时记忆网络的提出，解决了这一问题。说实在的我现在尚不清楚我们目前的目标需求为什么需要长时记忆。。。

LSTM

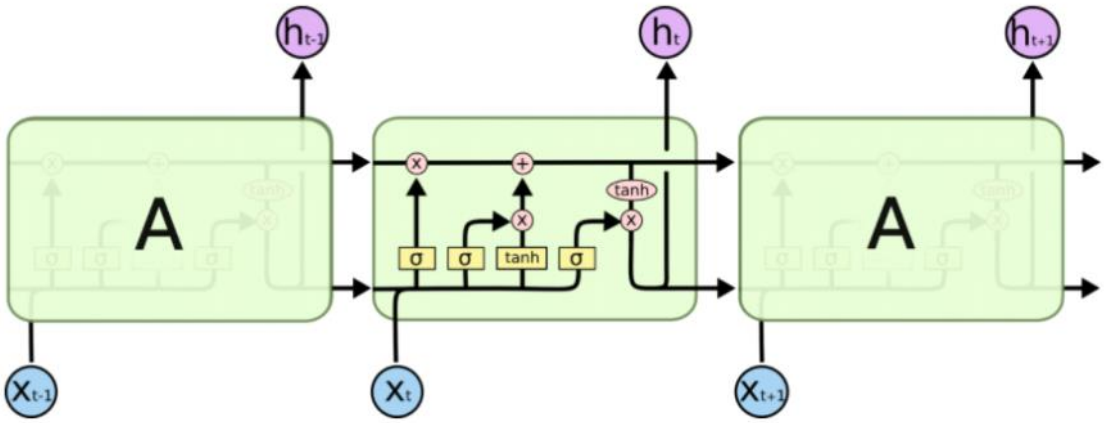
结构

首先回顾一下 RNN 模型的结构：

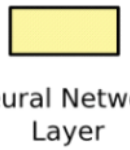


可以看到 RNN 在结合‘记忆’的方面的实现比较简单，其隐藏层只有一个状态，这个状态对短期记忆比较敏感，只是单纯地将时间上输入与结构上的输入简单地经过了 tanh 激活函数就直接输出了（回过头来看这一块的描述似乎存在问题）。因此导致的梯度消失已经在上一篇文章介绍了。

LSTM 对 RNN 的记忆结构进行了改进。为其增加了一个细胞状态，用来保存长期记忆，即结构图中上面的那一条线。下面会详细介绍。如图：



图中符号的解释：（事实上理解了 RNN 以后这幅图可以比较轻松地看懂）

- 

代表一个学习阶段的神经网路层，在数学上表现为若干个具体的更新方程。如：

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

- 

代表向量之间的线性运算。如加减乘除或是函数运算。

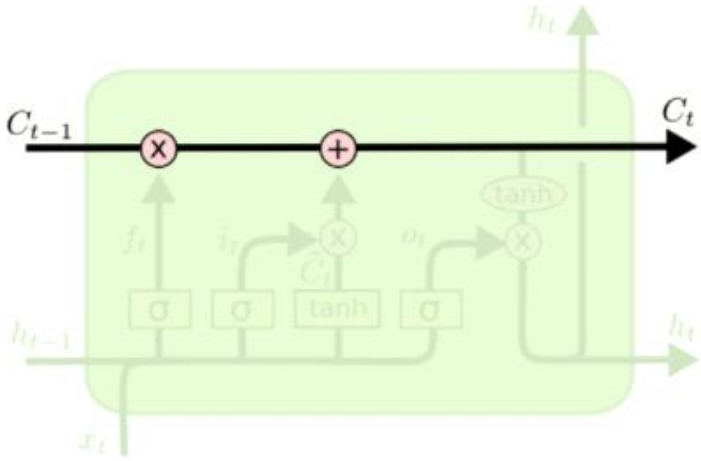
- 

代表向量的连接，如：

$$[h_{t-1}, x_t]$$

向前计算

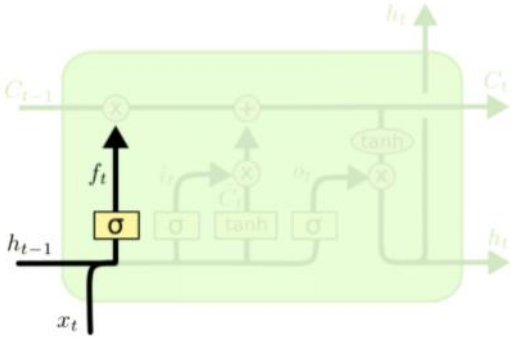
细胞状态



可以看到当前细胞状态（ Ct-1 ）做的其实是一个简单的线性交互（那些粉色的圈）。并将处理过后的值传递至下一个细胞状态（ Ct ）。

虽然细胞状态的传递看起来简单，其实真正关键的部分在于决定细胞状态更新以及输出的几个门结构。

遗忘门



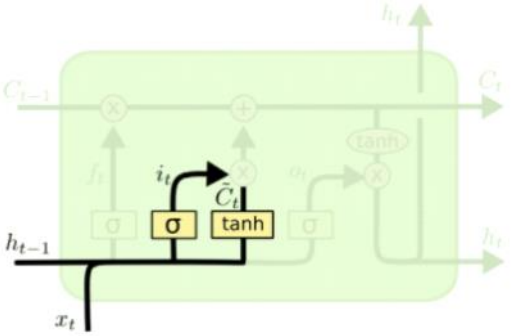
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

将记忆参数（自己起的名字，对应 RNN 中用于‘记忆’的参数） h_t-1 与实时输入参数 x_t 整合后加权 W_f·input + b_f 输入，最后由激活函数包裹传入细胞状态。这一步中输出的 f_t 介于 0 到 1 之间，后续与当前细胞状态 c_t-1 做乘法更新细胞状态，1 表示“完全保留”，0 表示“完全舍弃”。

功能

简单来说，这一步的功能就是**根据输入的记忆参数与实时输入参数结合权重生成‘遗忘’的比例。**需要注意的是对 c 的‘遗忘操作’是在细胞状态的流水线上进行的（即那个乘法粉红圆圈），这里尚未进行。

输入门



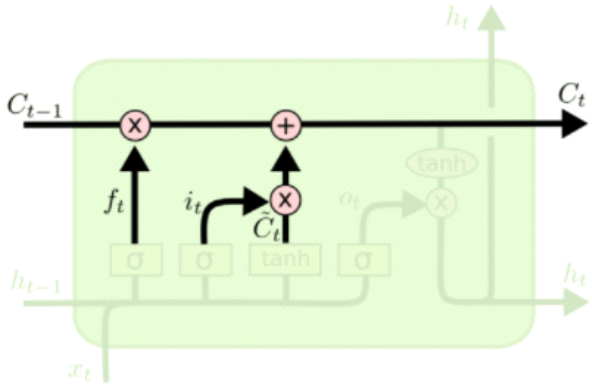
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

这里的 i_t 运算是根据输入参数（ h_t-1 与 x_t 即前面说的记忆参数与实时输入参数，后文用输入参数代指），得到当前的输入有多少需要被保留到细胞状态 c_t-1。而 C~_t 是表示的是经过输入参数得到的当前细胞状态结果（**候选值向量**，这个概念有点难懂，大概和后面又一次用到 tanh 函数的输出层的那条方程有关系）。

功能

这一步得到的**候选值向量 C~_t 与输入门的输出参数 i_t 做乘法即可得到最终需要输入到细胞状态的数据量。**最终在细胞状态的流水线上做相加操作就完成了输出门对细胞状态的更新。

更新细胞状态



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

正如前面的门的**功能部分**所说，

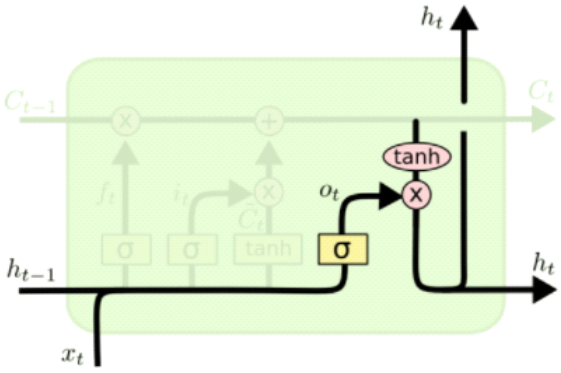
这是一条细胞状态更新的流水线，

首先进行‘遗忘更新操作’，与遗忘门输出 f_t 做乘法遗忘掉一些过去细胞状态中的无关因素。

再进行‘输入更新操作’，与输入门输出 i_t · C~_t 做加法，往细胞状态加入当前的记忆。

这样就生成了新的细胞状态。

输出门



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

额外地维护了一个细胞状态这一复杂参数，就是为了让其在长时记忆上产生做用，那么它当然会对神经元输出做出影响。

o_t 计算了在当前输入下需要将细胞状态中的哪些内容输出除去，h_t 利用 tanh 对细胞状态进行处理。

功能

> **将经处理的细胞状态 h_t 与 过滤细胞状态输出内容的参数 o_t 做乘法，得到最终的细胞输出。**

总结

遗忘门（forget gate）

它决定了上一时刻的单元状态 c_t-1 有多少保留到当前时刻 c_t

输入门（input gate）

它决定了当前时刻网络的输入 x_t 有多少保存到单元状态 c_t

输出门（output gate）

控制单元状态 c_t 有多少输出到 LSTM 的当前输出值 h_t

// 疑问：tanh 与 sigmoid 函数功能上究竟有何不同之处。应该是个问百度可以解决的问题。

反向传播

从前面的分析就能感觉到，LSTM 反向传播需要更新的梯度很多，这里就不做一一分析（其过程与经典的 BP 算法差别不大，一系列的链式法则与求导）。直接给出各梯度计算的方程。

W_{oh}、的W_{ih}、的W_{fh}、的W_{ch}

$$\frac{\partial E}{\partial W_{oh}} = \sum_{j=1}^{t} \delta_{o,j} \mathbf{h}_{j-1}^T$$

$$\frac{\partial E}{\partial W_{fh}} = \sum_{j=1}^{t} \delta_{f,j} \mathbf{h}_{j-1}^T$$

$$\frac{\partial E}{\partial W_{ih}} = \sum_{j=1}^{t} \delta_{i,j} \mathbf{h}_{j-1}^T$$

$$\frac{\partial E}{\partial W_{ch}} = \sum_{j=1}^{t} \delta_{\tilde{c},j} \mathbf{h}_{j-1}^T$$

$\mathbf{b}_f$、$\mathbf{b}_i$、$\mathbf{b}_c$、$\mathbf{b}_o$

$$\frac{\partial E}{\partial \mathbf{b}_o} = \sum_{j=1}^{t} \delta_{o,j}$$

$$\frac{\partial E}{\partial \mathbf{b}_i} = \sum_{j=1}^{t} \delta_{i,j}$$

$$\frac{\partial E}{\partial \mathbf{b}_f} = \sum_{j=1}^{t} \delta_{f,j}$$

$$\frac{\partial E}{\partial \mathbf{b}_c} = \sum_{j=1}^{t} \delta_{\tilde{c},j}$$

W_{fx}、W_{ix}、W_{cx}、W_{ox}

$$\frac{\partial E}{\partial W_{ox}} = \frac{\partial E}{\partial \mathbf{net}_{o,t}} \frac{\partial \mathbf{net}_{o,t}}{\partial W_{ox}} = \delta_{o,t} \mathbf{x}_t^T$$

$$\frac{\partial E}{\partial W_{fx}} = \frac{\partial E}{\partial \mathbf{net}_{f,t}} \frac{\partial \mathbf{net}_{f,t}}{\partial W_{fx}} = \delta_{f,t} \mathbf{x}_t^T$$

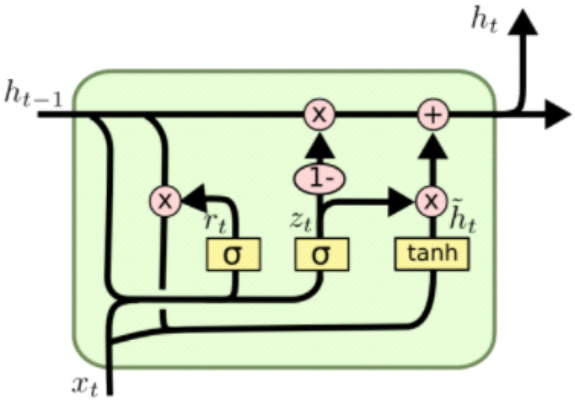
$$\frac{\partial E}{\partial W_{ix}} = \frac{\partial E}{\partial \mathbf{net}_{i,t}} \frac{\partial \mathbf{net}_{i,t}}{\partial W_{ix}} = \delta_{i,t} \mathbf{x}_t^T$$

$$\frac{\partial E}{\partial W_{cx}} = \frac{\partial E}{\partial \mathbf{net}_{\tilde{c},t}} \frac{\partial \mathbf{net}_{\tilde{c},t}}{\partial W_{cx}} = \delta_{\tilde{c},t} \mathbf{x}_t^T$$

GRU

这个算法是 LSTM 的一个流行的变体，其简化了 LSTM 的结构，将遗忘门与输入门整合成为了一个更新门（顾名思义，整合了细胞状态更新的过程），事实上它也整合了细胞状态这一概念与隐藏状态这一概念，将需要额外再细胞状态上做的流水线工作直接在隐藏状态上完成了。这一块先简单了解，后续需要用到了再深入学习。

结构与向前计算公式



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

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

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

末

今天拖延了，没有完成对 RNN 模型的 tensorflow 下的实现。因为在实际操作过程中发现还是需要挺多

有关 **LSTM** 与 **GRU** 这两种变种的知识，因此打算先补补课。争取尽快知识丰度能够支持后续针对市场法的整体架构有一定详细了解了以后模型建设马上开始动工。

参考博客

https://www.cnblogs.com/wangduo/p/6773601.html

https://www.jianshu.com/p/dcec3f07d3b5