

深度学习理论与实践

第7课：循环神经网络 RNN

主讲人 郑元春

中科院大数据挖掘与知识管理重点实验室
中科院虚拟经济与数据科学研究中心
从事机器学习与自然语言处理相关研究





RNN



RNN推导



LSTM



GRU



代码讲解

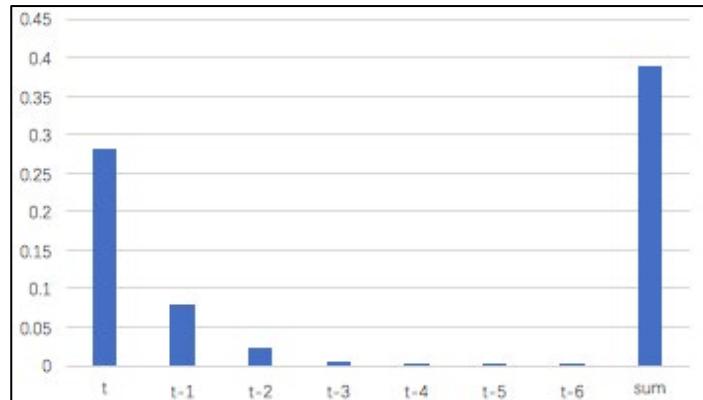


3. LSTM

RNN训练梯度消失问题

右边的图显示的是在某轮训练中，各个时刻的梯度以及最终的梯度之和。

我们就可以看到，从 $t-3$ 时刻开始，梯度已经几乎减少到0了。那么，从这个时刻开始再往之前走，得到的梯度（几乎为零）就不会对最终的梯度值有任何贡献，这就相当于无论 $t-3$ 时刻之前的网络状态 h 是什么，在训练中都不会对权重数组 W 的更新产生影响，也就是网络事实上已经忽略了 $t-3$ 时刻之前的状态。这就是原始RNN无法处理长距离依赖的原因。



既然找到了问题的原因，那么我们就能解决它。从问题的定位到解决，科学家们大概花了7、8年时间。终于有一天，Hochreiter和Schmidhuber两位科学家发明出**长短时记忆网络LSTM**，一举解决这个问题。

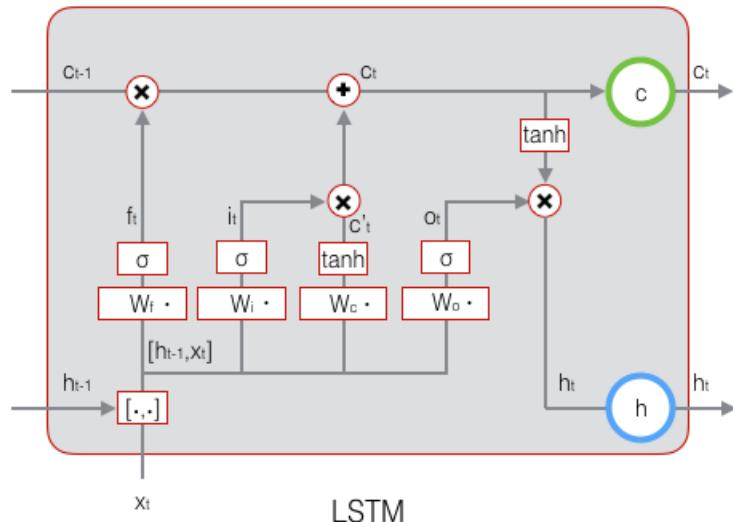
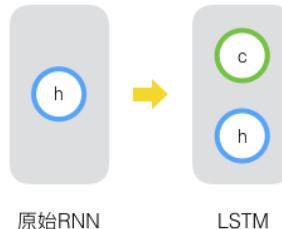


3. LSTM

基本单元

不管是LSTM，还是GRU，都指的的网络中某个神经单元的结构和传统的神经结构不同，并不是神经网络的结构不同。

其实，**长短时记忆网络**的思路比较简单。原始RNN的隐藏层只有一个状态，即 h ，它对于短期的输入非常敏感。那么，假如我们再增加一个状态，即 c ，让它来保存长期的状态，那么问题不就解决了么？如下图所示：



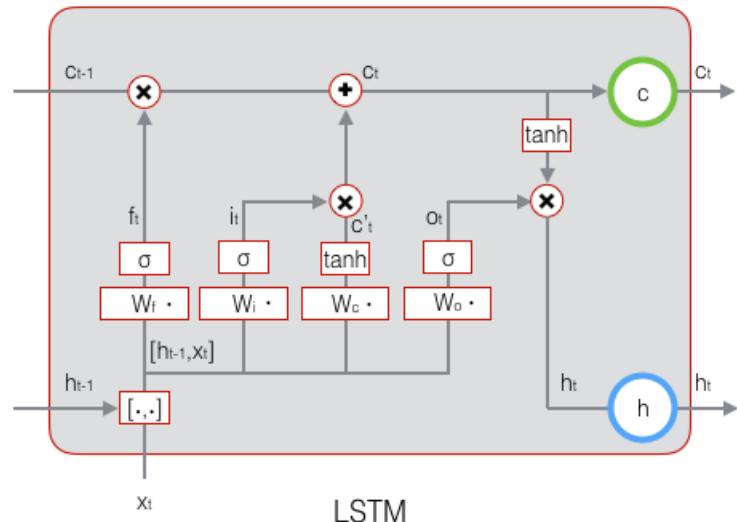
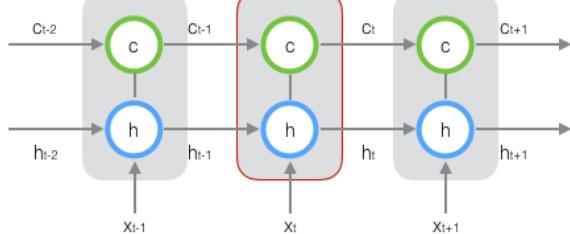


3. LSTM

基本单元

不管是LSTM，还是GRU，都指的的网络中某个神经单元的结构和传统的神经结构不同，并不是神经网络的结构不同。

新增加的状态 c ，称为**单元状态(cell state)**。我们把上图按照时间维度展开：





3. LSTM

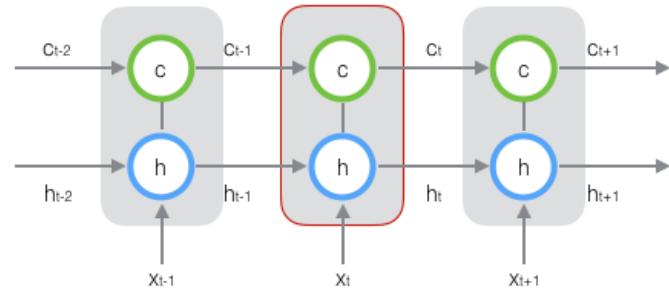
右图仅仅是一个示意图，我们可以看出，在 t 时刻：

LSTM的输入有三个：

- 当前时刻网络的输入值 x_t
 - 上一时刻LSTM的输出值 h_{t-1}
 - 上一时刻的单元状态 c_{t-1}

LSTM的输出有两个：

- 当前时刻LSTM输出值 h_t
 - 当前时刻的单元状态 c_t



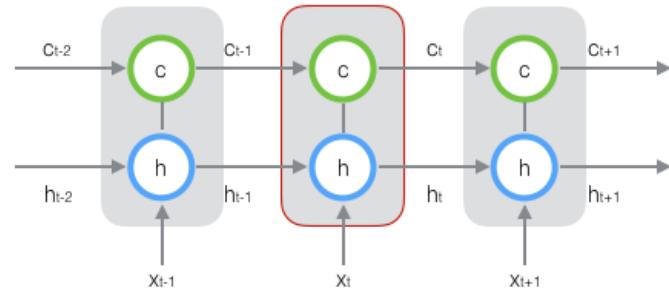


3. LSTM

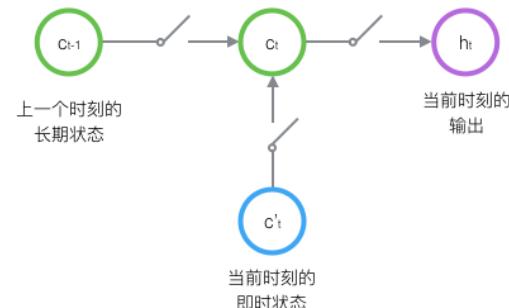
LSTM的关键，就是怎样控制长期状态 c 。在这里，LSTM的思路是使用三个控制开关。

- 第一个开关，负责控制继续保存长期状态
- 第二个开关，负责控制把即时状态输入到长期状态
- 第三个开关，负责控制是否把长期状态 c 作为当前的LSTM的输出。

三个开关的作用如右图所示：



长期状态 c 的控制

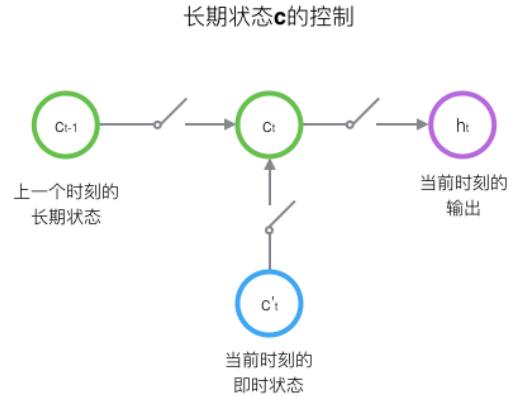




3. LSTM

前面描述的开关是怎样在算法中实现的呢？这就用到了**门（gate）**的概念。门实际上就是一层**全连接层**，它的输入是一个向量，输出是一个0到1之间的实数向量。假设 W 是门的权重向量， b 是偏置项，那么门可以表示为：

$$g(\mathbf{x}) = \sigma(W\mathbf{x} + \mathbf{b})$$



门的使用，就是用门的输出值按元素乘以我们需要控制的那个向量。因为门的输出是0到1之间的实数向量，那么，当门输出为0时，任何向量与之相乘都会得到0向量，这就相当于啥都不能通过；输出为1时，任何向量与之相乘都不会有任何改变，这就相当于啥都可以通过。因为（也就是sigmoid函数）的值域是(0,1)，所以门的状态都是半开半闭的。



3. LSTM

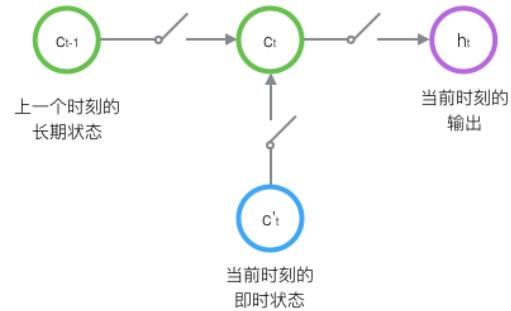
LSTM用两个门来控制单元状态c的内容

遗忘门(Forget gate): 它决定了上一时刻的单元状态有多少保留到当前时刻

输入门(Input gate): 它决定了当前时刻网络的输入有多少保存到单元状态

输出门(Output gate): 来控制单元状态有多少输出到LSTM的当前输出值

长期状态c的控制





3. LSTM

LSTM用两个门来控制单元状态c的内容

遗忘门(Forget gate): 它决定了上一时刻的单元状态有多少保留到当前时刻

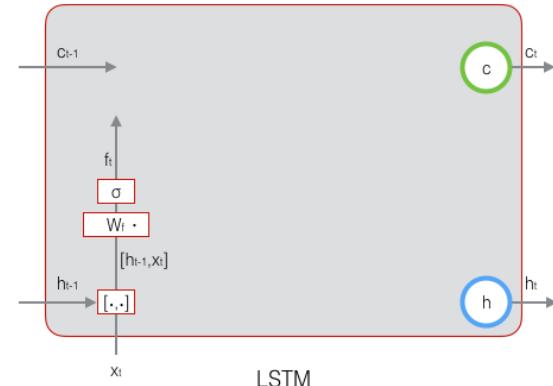
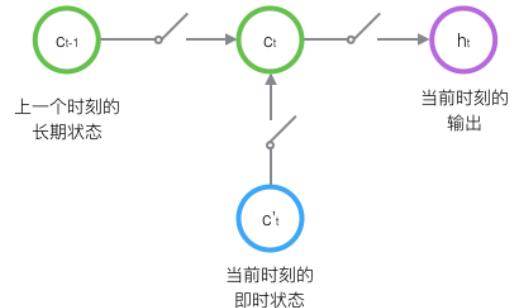
输入门(Input gate): 它决定了当前时刻网络的输入有多少保存到单元状态

输出门(Output gate): 来控制单元状态有多少输出到LSTM的当前输出值

$$\mathbf{f}_t = \sigma(W_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

上式中, W_f 是遗忘门的权重矩阵, $[\mathbf{h}_{t-1}, \mathbf{x}_t]$ 表示把两个向量连接成一个更长的向量, \mathbf{b}_f 是遗忘门的偏置项, σ 是sigmoid函数。

长期状态c的控制





3. LSTM

LSTM用两个门来控制单元状态c的内容

遗忘门(Forget gate): 它决定了上一时刻的单元状态有多少保留到当前时刻

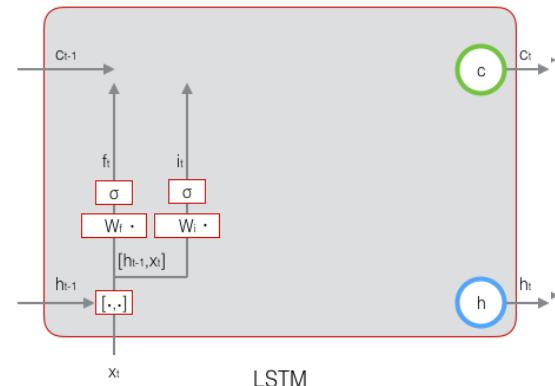
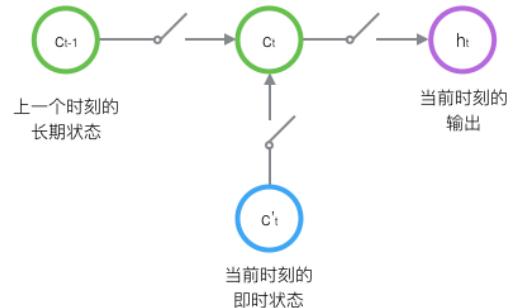
输入门(Input gate): 它决定了当前时刻网络的输入有多少保存到单元状态

输出门(Output gate): 来控制单元状态有多少输出到LSTM的当前输出值

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

上式中, W_i 是输入门的权重矩阵, $[h_{t-1}, x_t]$ 还是同样的拼接向量, b_i 是遗忘门的偏置项, σ 也是sigmoid函数。

长期状态c的控制





3. LSTM

LSTM用两个门来控制单元状态c的内容

遗忘门(Forget gate): 它决定了上一时刻的单元状态有多少保留到当前时刻

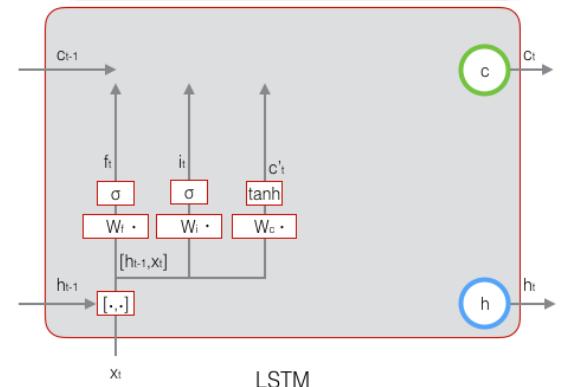
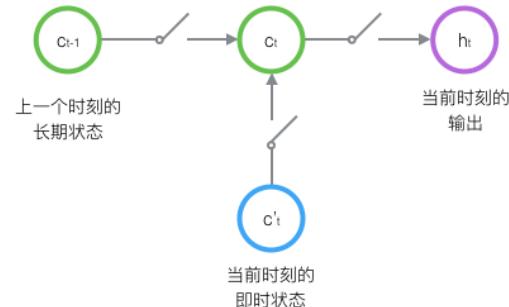
输入门(Input gate): 它决定了当前时刻网络的输入有多少保存到单元状态

输出门(Output gate): 来控制单元状态有多少输出到LSTM的当前输出值

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

当前输入的单元状态，和RNN中的一致。这里将激活函数换成了tanh函数

长期状态c的控制





3. LSTM

LSTM用两个门来控制单元状态c的内容

遗忘门(Forget gate): 它决定了上一时刻的单元状态有多少保留到当前时刻

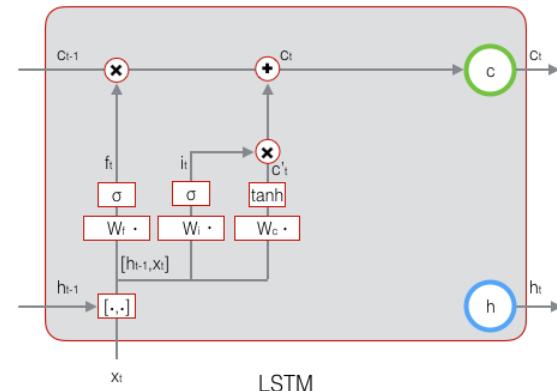
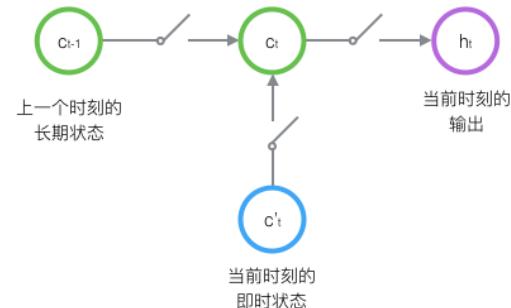
输入门(Input gate): 它决定了当前时刻网络的输入有多少保存到单元状态

输出门(Output gate): 来控制单元状态有多少输出到LSTM的当前输出值

$$\mathbf{c}_t = f_t \circ \mathbf{c}_{t-1} + i_t \circ \tilde{\mathbf{c}}_t$$

当前时刻的单元状态，上一次单元状态通过遗忘门，当前输入的单元状态通过输入门，再加和在一起。

长期状态c的控制





3. LSTM

LSTM用两个门来控制单元状态c的内容

遗忘门(Forget gate): 它决定了上一时刻的单元状态有多少保留到当前时刻

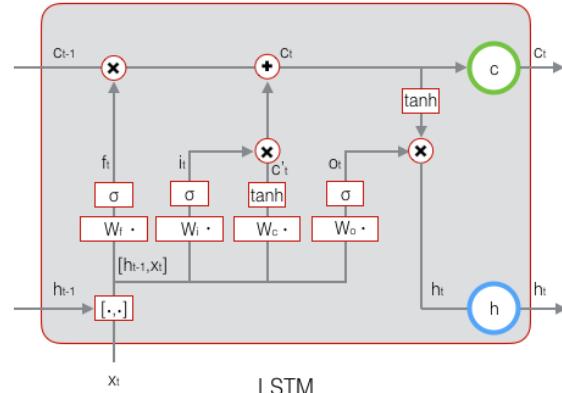
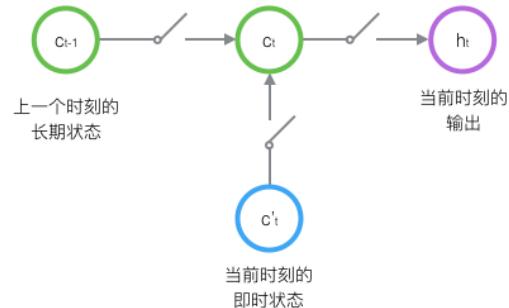
输入门(Input gate): 它决定了当前时刻网络的输入有多少保存到单元状态

输出门(Output gate): 来控制单元状态有多少输出到LSTM的当前输出值

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

LSTM的最终时刻的输出，是由输出门和单元状态同时确定的：

长期状态c的控制





3. LSTM

LSTM的训练方法：

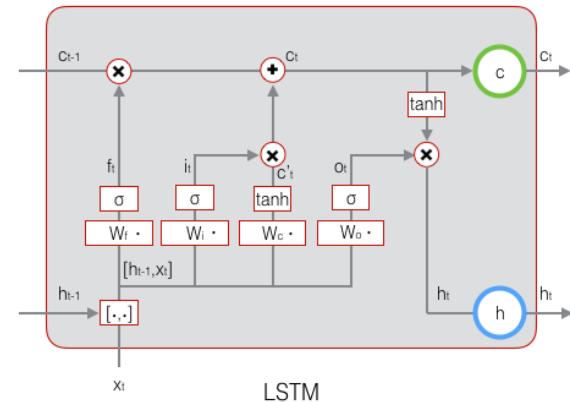
- 前向计算每个神经元的输出值，对于LSTM来说，即 f_t 、 i_t 、 c_t 、 o_t 、 h_t 五个向量的值。
- 反向计算每个神经元的**误差项值**。与循环神经网络一样，LSTM误差项的反向传播也是包括两个方向：一个是沿时间的反向传播，即从当前t时刻开始，计算每个时刻的误差项；一个将误差项向上一层传播。
- 根据相应的误差项，计算每个权重的梯度。

$$\sigma(z) = y = \frac{1}{1 + e^{-z}}$$

$$\sigma'(z) = y(1 - y)$$

$$\tanh(z) = y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\tanh'(z) = 1 - y^2$$

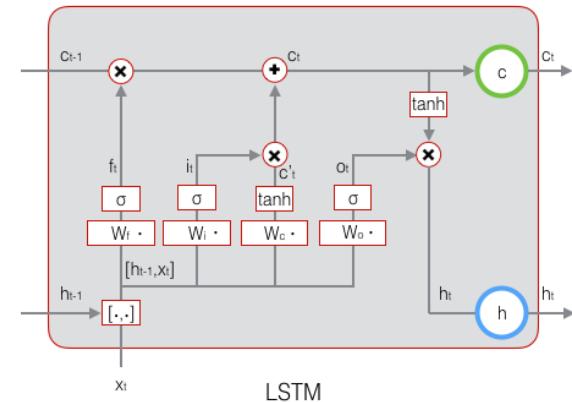




3. LSTM

LSTM的训练方法：

- 前向计算每个神经元的输出值，对于LSTM来说，即 f_t 、 i_t 、 c_t 、 o_t 、 h_t 五个向量的值。
- 反向计算每个神经元的**误差项值**。与循环神经网络一样，LSTM误差项的反向传播也是包括两个方向：一个是沿时间的反向传播，即从当前t时刻开始，计算每个时刻的误差项；一个是在将误差项向上一层传播。
- 根据相应的误差项，计算每个权重的梯度。



LSTM需要学习的参数共有8组，分别是：遗忘门的权重矩阵 W_f 和偏置项 b_f 、输入门的权重矩阵 W_i 和偏置项 b_i 、输出门的权重矩阵 W_o 和偏置项 b_o ，以及计算单元状态的权重矩阵 W_c 和偏置项 b_c 。因为权重矩阵的两部分在反向传播中使用不同的公式，因此在后续的推导中，4个权重矩阵都将被写为分开的两个矩阵： W_{fh} 、 W_{fx} 、 W_{ih} 、 W_{ix} 、 W_{oh} 、 W_{ox} 、 W_{ch} 、 W_{cx} 。



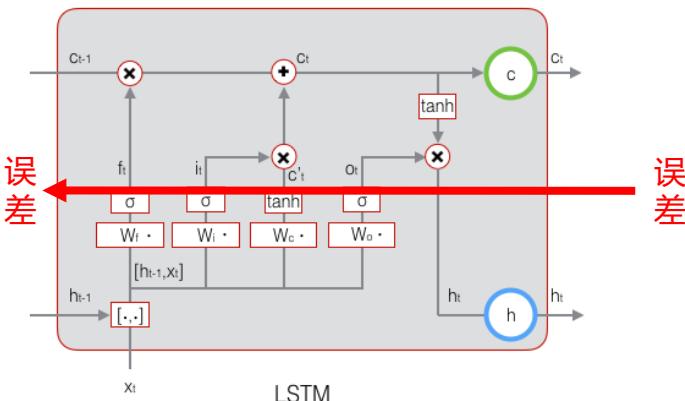
3. LSTM

LSTM的训练方法：

在 t 时刻，LSTM的输出值为 h_t ，定义 t 时刻的误差项 δ_t ：

$$\delta_t \stackrel{\text{def}}{=} \frac{\partial E}{\partial \mathbf{h}_t}$$

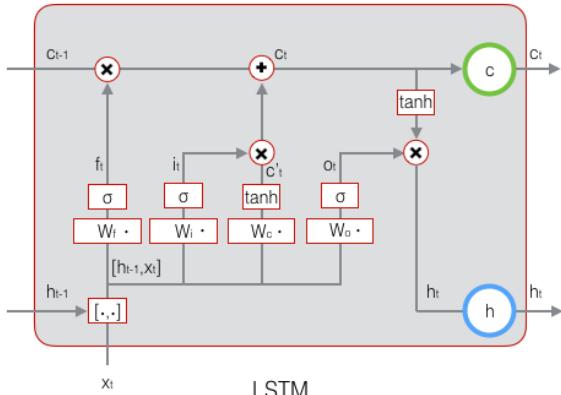
注意，和前面不同，我们这里假设误差项是损失函数对输出值的导数，而不是对加权输入 net_t^l 的导数。因为LSTM有四个加权输入，分别对应 f_t 、 i_t 、 c_t 、 o_t ，我们希望往上一层传递一个误差项而不是四个。但我们仍然需要定义出这四个加权输入，以及他们对应的误差项。





3. LSTM

注意，和前面不同，我们这里假设误差项是损失函数对输出值的导数，而不是对加权输入 net_t^i 的导数。因为LSTM有四个加权输入，分别对应 f_t 、 i_t 、 c_t 、 o_t ，我们希望往上一层传递一个误差项而不是四个。但我们仍然需要定义出这四个加权输入，以及他们对应的误差项。



$$\mathbf{net}_{f,t} = W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f$$

$$= W_{fh}\mathbf{h}_{t-1} + W_{fx}\mathbf{x}_t + \mathbf{b}_f$$

$$\mathbf{net}_{i,t} = W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i$$

$$= W_{ih}\mathbf{h}_{t-1} + W_{ix}\mathbf{x}_t + \mathbf{b}_i$$

$$\mathbf{net}_{\tilde{c},t} = W_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c$$

$$= W_{ch}\mathbf{h}_{t-1} + W_{cx}\mathbf{x}_t + \mathbf{b}_c$$

$$\mathbf{net}_{o,t} = W_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o$$

$$= W_{oh}\mathbf{h}_{t-1} + W_{ox}\mathbf{x}_t + \mathbf{b}_o$$

$$\delta_{f,t} \stackrel{\text{def}}{=} \frac{\partial E}{\partial \mathbf{net}_{f,t}}$$

$$\delta_{i,t} \stackrel{\text{def}}{=} \frac{\partial E}{\partial \mathbf{net}_{i,t}}$$

$$\delta_{\tilde{c},t} \stackrel{\text{def}}{=} \frac{\partial E}{\partial \mathbf{net}_{\tilde{c},t}}$$

$$\delta_{o,t} \stackrel{\text{def}}{=} \frac{\partial E}{\partial \mathbf{net}_{o,t}}$$



3. LSTM

② LSTM权重更新

对于 W_{fh} 、 W_{oh} 、 W_{ih} 、 W_{ch} 的权重梯度，我们知道它的梯度是各个时刻梯度之和，我们首先求出它们在t时刻的梯度，然后再求出他们最终的梯度。

我们已经求得了误差项 δ_{ot} 、 δ_{ft} 、 δ_{it} 、 δ_{ct} ，很容易求出t时刻 W_{fh} 、 W_{oh} 、 W_{ih} 、 W_{ch} ：

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

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

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

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

$$\begin{aligned} \frac{\partial E}{\partial W_{oh,t}} &= \frac{\partial E}{\partial \text{net}_{o,t}} \frac{\partial \text{net}_{o,t}}{\partial W_{oh,t}} \\ &= \delta_{o,t} \mathbf{h}_{t-1}^T \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial W_{fh,t}} &= \frac{\partial E}{\partial \text{net}_{f,t}} \frac{\partial \text{net}_{f,t}}{\partial W_{fh,t}} \\ &= \delta_{f,t} \mathbf{h}_{t-1}^T \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial W_{ih,t}} &= \frac{\partial E}{\partial \text{net}_{i,t}} \frac{\partial \text{net}_{i,t}}{\partial W_{ih,t}} \\ &= \delta_{i,t} \mathbf{h}_{t-1}^T \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial W_{ch,t}} &= \frac{\partial E}{\partial \text{net}_{\tilde{c},t}} \frac{\partial \text{net}_{\tilde{c},t}}{\partial W_{ch,t}} \\ &= \delta_{\tilde{c},t} \mathbf{h}_{t-1}^T \end{aligned}$$



3. LSTM

② LSTM权重更新

对于 b_f 、 b_i 、 b_c 、 b_o 的权重梯度，也是将各个时刻的梯度加在一起：

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

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

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

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

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{b}_{o,t}} &= \frac{\partial E}{\partial \mathbf{net}_{o,t}} \frac{\partial \mathbf{net}_{o,t}}{\partial \mathbf{b}_{o,t}} \\ &= \delta_{o,t} \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{b}_{f,t}} &= \frac{\partial E}{\partial \mathbf{net}_{f,t}} \frac{\partial \mathbf{net}_{f,t}}{\partial \mathbf{b}_{f,t}} \\ &= \delta_{f,t} \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{b}_{i,t}} &= \frac{\partial E}{\partial \mathbf{net}_{i,t}} \frac{\partial \mathbf{net}_{i,t}}{\partial \mathbf{b}_{i,t}} \\ &= \delta_{i,t} \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{b}_{c,t}} &= \frac{\partial E}{\partial \mathbf{net}_{c,t}} \frac{\partial \mathbf{net}_{c,t}}{\partial \mathbf{b}_{c,t}} \\ &= \delta_{c,t} \end{aligned}$$



3. LSTM

③ 梯度问题

LSTM并不能完全的解决梯度问题，仅仅知识缓解，原始的LSTM是没有遗忘们的，因此 C_t 的更新如下：

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

由此可见 C_t 对 C_{t-1} 的偏导为常数1，因此有效的缓解了LSTM的梯度消失问题。但是现在用的LSTM加入了遗忘门，使得 C_t 的更新如下：

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

由于 f_t 是由sigmoid函数得来的，而sigmoid函数的范围是 (0,1) ，因此加入了遗忘门之后，可以说又将LSTM的梯度消失的问题有加重了。但是此时的梯度消失，可以解释为网络故意遗忘前面部分。



3. LSTM

③ 梯度问题

- RNN中总的梯度不会消失(可参考35页图)。即便梯度越传越弱，那也只是远距离的梯度消失，由于近距离的梯度不会消失，所有梯度之和便不会消失。RNN 所谓梯度消失的真正含义是，梯度被近距离梯度主导，导致模型难以学到远距离的依赖关系。
- LSTM 中梯度的传播有很多条路径，这条路径上只有逐元素相乘和相加的操作，梯度流最稳定；但是其他路径上梯度流与普通 RNN 类似，照样会发生相同的权重矩阵反复连乘。
- LSTM 刚提出时没有遗忘门，或者说相当于~~，~~，这时候在直接相连的短路路径上可以无损地传递，从而这条路径上的梯度畅通无阻，不会消失。类似于 ResNet 中的残差连接。
- 但是在其他路径上，LSTM 的梯度流和普通 RNN 没有太大区别，依然会爆炸或者消失。由于总的远距离梯度 = 各条路径的远距离梯度之和，即便其他远距离路径梯度消失了，只要保证有一条远距离路径梯度不消失，总的远距离梯度就不会消失（正常梯度 + 消失梯度 = 正常梯度）。因此 LSTM 通过改善一条路径上的梯度问题拯救了总体的远距离梯度。
- 同样，因为总的远距离梯度 = 各条路径的远距离梯度之和，高速公路上梯度流比较稳定，但其他路径上梯度有可能爆炸，此时总的远距离梯度 = 正常梯度 + 爆炸梯度 = 爆炸梯度，因此 LSTM 仍然有可能发生梯度爆炸。不过，由于 LSTM 的其他路径非常崎岖，和普通 RNN 相比多经过了很多次激活函数（导数都小于 1），因此 LSTM 发生梯度爆炸的频率要低得多。实践中梯度爆炸一般通过梯度裁剪来解决。



RNN



RNN推导



LSTM



GRU



代码讲解

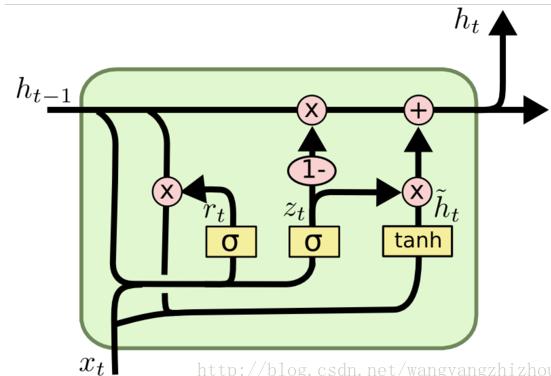


4. GRU

LSTM的结构比较复杂，实现了三个门计算，即遗忘门，
输入门和输出门。

而GRU模型只剩下两个门了，分别为更新门和重置门，
即图中的 z_t 和 r_t 。

- 更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。
- 重置门用于控制忽略前一时刻的状态信息的程度，重置门的值越小说明忽略得越多。



<http://blog.csdn.net/wangyangzhizhou>



4. GRU

LSTM的结构比较复杂，实现了三个门计算，即遗忘门，
输入门和输出门。

而GRU模型只剩下两个门了，分别为更新门和重置门，
即图中的 z_t 和 r_t 。

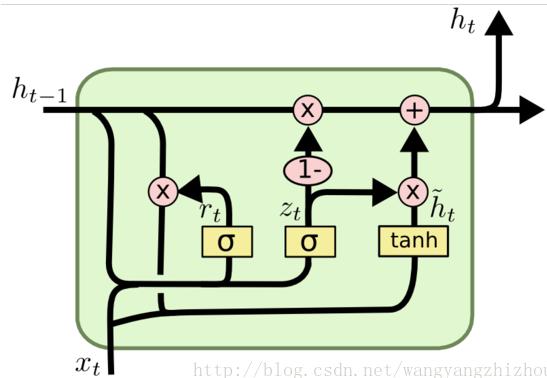
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

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

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

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

$$y_t = \sigma(W_o \cdot h_t)$$



<http://blog.csdn.net/wangyangzhizhou>

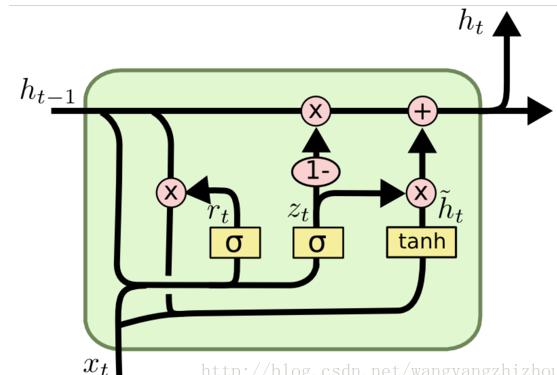
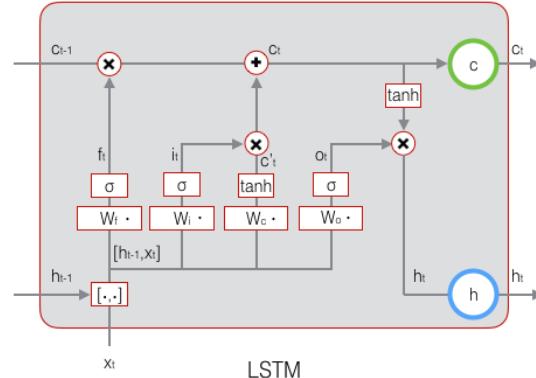


4. GRU

LSTM和GRU之间并没有明显的优胜者.

GRU具有较少的参数，所以训练的速度快，而且所需要的实验样本也比较少。

LSTM具有较多的参数，并且适合具有大量样本的情况，可能会获得较优的模型。





RNN



GRU



RNN推导



LSTM



代码讲解