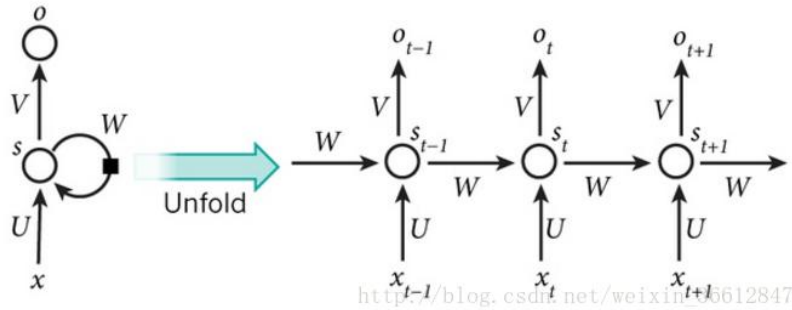
LSTM 深入（长短时记忆网络）

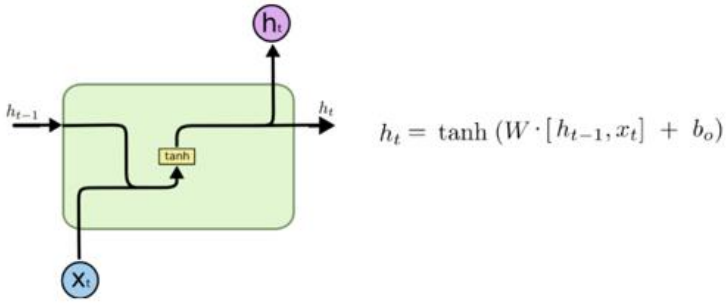
前身：RNN

https://zhuanlan.zhihu.com/p/32085405



每一个圆点代表序列每个节点的训练单元，注意每个圆点都有相同的权值 w 和 u，--》每个序列节点训练更新相同的 w 和 u，权值是共享的状态。每一个序列节点都 lstm 单元都接受这个序列节点的 data 和之前得到的状态值。

单个 RNN 单元结构，计算公式：



当前层各个参数含义：

x_t ：当前时刻的输入

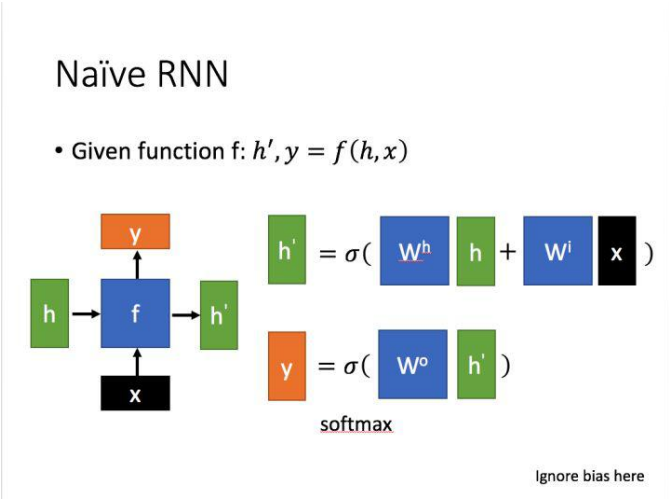
H_{t-1}：上一时刻的输出

C_{t-1}：上一时刻的输出

W ：需要训练的参数

B ：需要训练的参数

单个 RNN 每个时刻输入输出 和 向量维度：



RNN：每一时刻输入输出

$h_t = \tanh(W \cdot [h_{t-1}, x_t] + b_o)$

x_t ：[batch_size , input_size]

H_{t-1} ：[batch_size , Hidden_size]

W ：[input_size + Hidden_size , Hidden_size]

B ：[Hidden_size]

$[h_{t-1}, x_t] = [x_t , H_{t-1}]$ ：

得到的 shape 为：[batch_size , input_size + Hidden_size]

$[h_{t-1}, x_t] \cdot W$ ：

矩阵相乘计算后的到的 Shape 为：[batch_size , Hidden_size]

$[h_{t-1}, x_t] \cdot W + b_o$ ：

加上 B 计算后得到的 H_t shape 为：

[batch_size , Hidden_size]

rnn 容易出现的梯度消散，爆炸的问题！！

LSTM 深入：

https://www.jianshu.com/p/dcec3f07d3b5（详细计算）

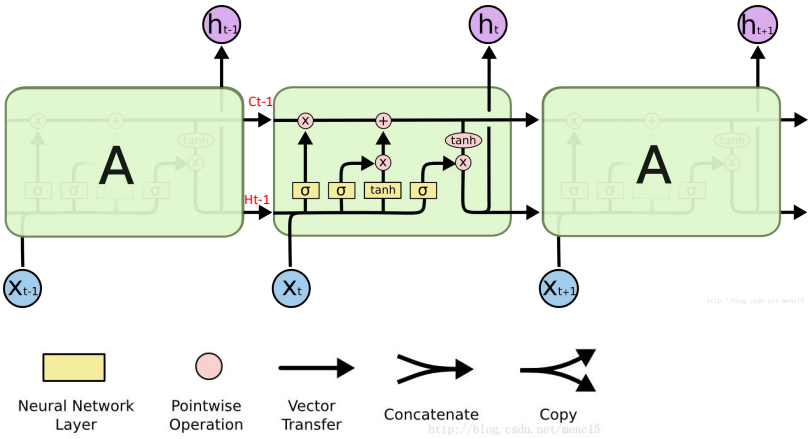
https://blog.csdn.net/menc15/article/details/71271566

<1>设计初衷： 设计初衷是希望能够解决神经网络中的长期依赖问题，让记住长期信息成为神经网络的默认行为，而不是需要很大力气才能学会

<2>基本原理：

增加了 rnn 单元的复杂度，更仔细地建模，有了更多限制条件，使得训练变得更加轻松，解决了 rnn 容易出现的梯度消散的问题。

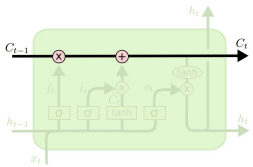
<3>LSTM 单元结构：



关键：两个传输状态(cell state,hidden state)：相比 RNN 只有一个传递状态 ht

(1)单元状态（cell state）Ct，即图中 LSTM 单元上方从左贯穿到右的水平线，它像是传送带一样，将信息从上一个单元传递到下一个单元，和其他部分只有很少的线性的相互作用 **（Ct--类似于 RNN 里的 ht）**

---->对于传递下去的 Ct **主要是用来保存先前节点的数据的，之前节点输入向量的叠加，改变得很慢**，通常输出的 Ct 是上一个状态传过来的 Ct-1 加上一些数值。解释：主要是用来保存节点传递下来的数据的，每次传递会对某些维度进行“忘记”并且会加入当前节点所包含的内容，总的来说还是用来保存节点的信息，改变相对较小

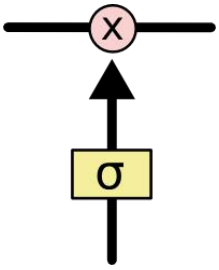


(2) hidden state Ht

而 **Ht 则在不同节点下差别往往会有很大，取决于当前单元的输入 Xt，Ht-1，** 解释：Ht 主要是为了和当前输入组合来获得门控信号，对于不同的当前输入，传递给下一个状态的 Ht 区别也会较大

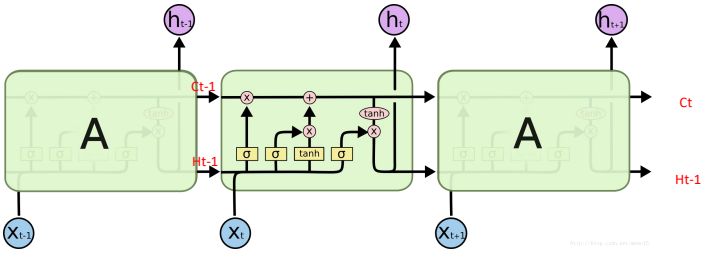
LSTM 通过“门”（gate）来控制丢弃或者增加信息，从而实现遗忘或记忆的功能。

“**门 gate**”是一种使信息选择性通过的结构，由***一个 sigmoid 函数***和***一个点乘***操作组成。sigmoid 函数的输出值在[0,1]区间，*0 代表完全丢弃，1 代表完全通过*。一个 LSTM 单元有三个这样的门，分别是遗忘门（forget gate）、输入门（input gate）、输出门（output gate）。



<4>LSTM 单元部件分析：

输入输出，向量维度：



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

x_t ：[batch_size， input_size]

H_{t-1} : [batch_size，Hidden_size]

C_{t-1} : [batch_size，Hidden_size]

[Wf,wi,wc,wo]== N : [input_size + Hidden_size，4 * Hidden_size]

[Bf,Bi,Bc,Bo]= B ： [4 * Hidden_size]

x_t ：[batch_size， input_size]

H_{t-1} : [batch_size，Hidden_size]

C_{t-1} : [batch_size，Hidden_size]

w_L ：[input_size + Hidden_size，4 * Hidden_size]

B_L ： [4 * Hidden_size]

1，$[h_{t-1}, x_t] \cdot W + b_o$ ：直接计算四个，如下：

[batch_size, input_size + Hidden_size]* [input_size +Hidden_size，4 * Hidden_size]

【f_t，i_t，$\tilde{C}_t$，o_t】 = [batch_size，4 * Hidden_size]

2，然后分成四份，分别得到：f_t，i_t，$\tilde{C}_t$，o_t

Shape 的大小都是：[batch_size，Hidden_size]

3，计算 $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ ，此处的乘为对应元素相乘

C_t= multiply（C_{t-1},sigmoid（f_t））+ multiply（tanh($\tilde{C}_t$),sigmoid(i_t)）

计算后：C_t 的 shape 为：[batch_size, Hidden_size]

4，计算 $h_t = o_t * \tanh\left(C_t\right)$ ，此处依然为对应元素相乘

h_t = multiply（sigmoid（o_t），C_t）

计算后的 shape 为：[batch_size, Hidden_size]

单元输入： Ct-1,Ht-1,Xt

单元结构：

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`

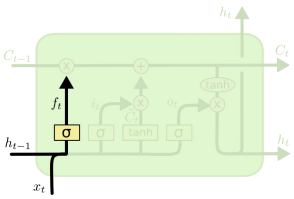
【注】

*****计算基本对象：[ht-1,xt]拼接表示：过去+现在输入细胞状态表示:[batch,hid+input size]

--》计算 遗忘程度，记忆程度的对象，输出的基本对象

（1） f:遗忘门（forget gate）： ----***对上一个节点传进来的输入每一项被遗忘的程度 ： 0-1***：之前细胞状态中哪些信息丢弃和保留---》**主要学习上一个细胞的遗忘程度**

决定***对上一个节点传进来的输入***有多少保存到当前时刻 Ct 【C_t-1 与 C_t】



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

W：[input+hid ,4*hid]-->Wf=[input+hid,hid]

B=[4*hid] --》Bf:[hid]

[ht-1,xt]=[batch,hid]+[batch,input]=[batch,hid+input]：

-->**每个 Tokens**：被之前和现在信息[1*hid+input]向量共同表示

W*[ht-1,xt]=[batch,hid+input]*[input+hid,hid]=[batch,hid]

W*[ht-1,xt]+bf :[hid]-->给每一列广播--》ft=【batch,hid】

-->每个 Token: Token 里每个元素都做了 W* +b-->sigimod-->0--1 值

输入：输入上一个单元输出 ht-1，本单元的输入 xt.

过程：合并 ht-1，Xt，-->点乘 Wf，加 bf-->sigimod->[0..1]---》ft-1

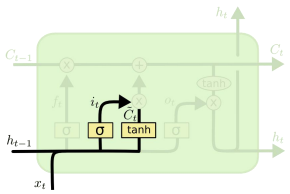
输出：***为 ht-1 中的每一项产生一个在[0,1]内的值***，来控制*上一单元状态中每一项(num=hid)被遗忘的程度 ： 0-1*。

（2） I:输入门（input gate）：当前结点的输入有选择性的有多少保存到单元状态 Ct 【X_t 与 C_t】

---主要学习现在输入 Xt 的记忆程度，这个阶段将这个阶段的输入有选择性地进行“记忆”。主要是会对输入 Xt

进行选择记忆

---sigimoid()和一个 tanh()配合控制有哪些新信息被加入(为下一步更新细胞状态 Ct 做准备)



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

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

：***过去现在信息输入拼接[ht-1, xt]***每一项被更新进入 Ct 的***程度***：Xt 每一项输出 0-1 值程度

：用来描述当前输入的单元状态

与 Ft 同理：It,Ct~:[batch,hid]

输入：输入上一个单元输出 ht-1，本单元的输入 xt.

过程：sigimoid,tanh 同样输入，不同目的

Sigimoid-->**it：**当前输入 Xt 每一项被更新进入 Ct 的程度：决定我们将更新哪些值，每一项生成一个 0-1 的值

tanh-->一个新的候选向量 **Ct~**(用来描述当前输入的单元状态)，

最后：结合 it,Ct~ 更新细胞状态（见下）【C_t-1--->C_t】

输出：

It:当前输入 Xt 每一项被更新进入 Ct 的程度

Ct~ ：用来描述当前输入的单元状态

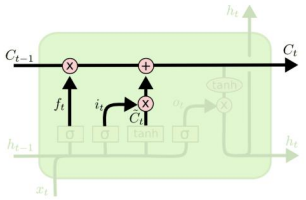
准备下一步更新细胞状态（C_t-1-->C_t）

(3) ***更新细胞状态（c_t-1 -->c_t）：****将旧单元状态 C_t-1 更新为新单元状态 C_t*

*----》****当前单元状态 Ct=上一个状态 Ht-1 保留的信息****(Ht-1 *每一项是否保留 Ft)+****当前状态 Xt 保留的信息****(当前状态表示 Ct~*每一项是否保留)=*

----前面的步骤已经决定了将会做什么，我们现在就是实际去完成

C_t= multiply (C_{t-1},sigmoid (f_t)) + multiply (tanh($\tilde{C}_t$),sigmoid(i_t))



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

Ft,,it,Ct~,Ct-1：[batch ,hid]

Ct=【batch,hid】

输入：

ft ：***对上一个节点传进来的输入每一项被遗忘的程度***

C_t-1：上一个细胞状态

it ：当前输入 Xt 每一项被更新进入 Ct 的程度，每一项生成一个 0-1 的值

C_t~:用来描述当前输入的单元状态

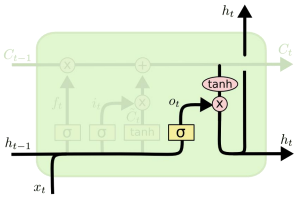
过程：

首先：(旧状态 C_t-1) * (上一个细胞状态每个项的遗忘程度 ft):*丢弃掉我们确定需要丢弃的信息*

然后：（当前输入 Xt 每一项被更新进入 Ct 的程度 it）*（C_t~用来描述当前输入的单元状态）：(决定更新那些值*候选向量 C_t~==根据缩放比例更新后新的候选向量 Ct)*：这就是新的候选值，根据我们决定更新每个状态的程度进行变化。*

最后：上一个细胞状态的遗留 + 更新状态

(4) O:输出门 (output gate) *:控制当前的单元状态 Ct 中过滤后有多少输出到 LSTM 当作当前输出值 ht* 【 Ct 与 ht】



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

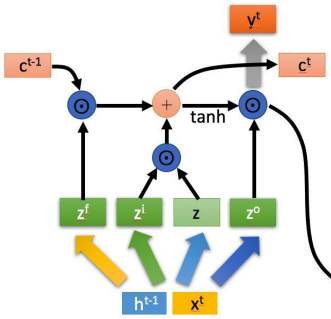
输入：

h_t-1 ,xt, Ct

输出：ht [batch,hid]

先将单元状态激活，输出门为其中每一项产生一个在[0,1]内的值，控制当前单元状态被过滤的程度

从 LSTM 整体网络看：



$$c^t = z^f \odot c^{t-1} + z^i \odot z$$
$$h^t = z^o \odot tanh(c^t)$$
$$y^t = \sigma(W' h^t)$$

输入：maxlen*【batch*embeddingDim:inputsize】（维度顺序不定）

---->两者通 xW+b 转换，其中 w ([embed_dim, hidden_size])。将输入维度 embed_dim 转化成了输出维度 hidden_size

输出：maxlen*【batch*hiddingsize】（维度顺序不定）

一般可以设定 Hidden Size=2*embeddingDim /embeddingDim（具体看效果）

【注】Tensorflow 的输出是 max*[batch*embeddingDim]的原因：

tensorflow 实现语言模型例子中是让 embed_dim 维度与 hidden_size 一致的

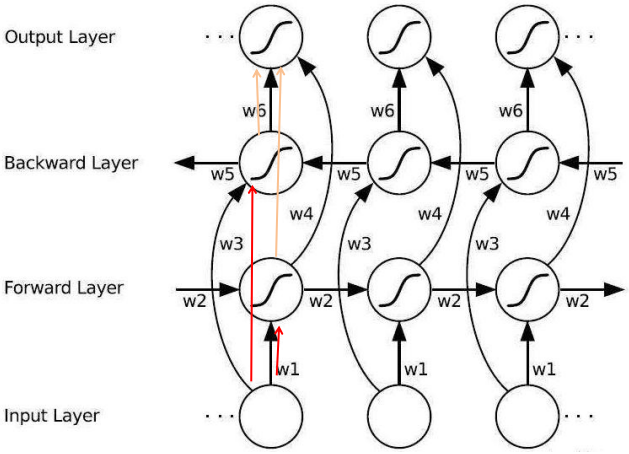
BiLSTM 双向

Forward:从 1 到 t 计算一遍：**batch*【maxlen,hid】**

Backward: 从 t 到 1 计算一遍：batch*[maxlen,hid]

最终输出：Concat(axis=-1): Batch*[maxlen,2*hid] OR plus batch*[maxlen,hid]

Concat/plus


Output Layer
Backward Layer
Forward Layer
Input Layer
w1
w2
w3
w4
w5
w6