

神经网络--RNN | LSTM | GRU

[[TOC]]

0. 资料网址:

- [核心机器之心的post](#)
- [三个神经网络的动图解释](#)
- [飞桨文档](#)
- [Deep Learning Book](#)
- [参考视频--可视化数学非常好的教学](#)
- [SVD分解的图解视频](#)
- [介绍LSTM](#)

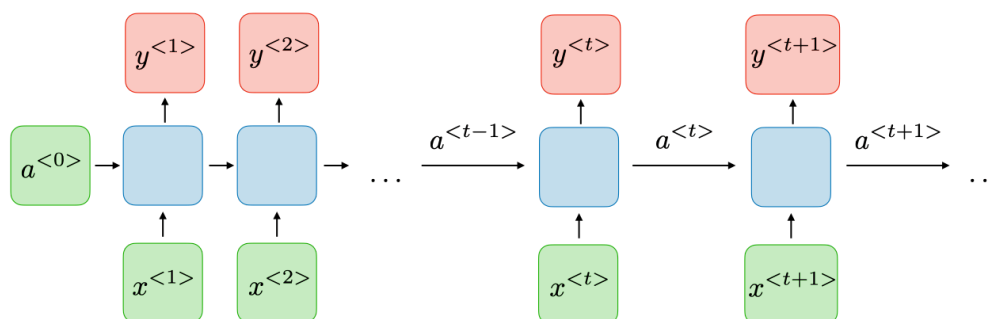
1. RNN

1.1 RNN的动机

- 需要处理变长的序列
- 要学习到输入的时间的依赖关系

1.2 RNN的结构和公式

结构



- 上述结构图参考(<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture>)

公式

RNN公式: 隐层 — 输入, 上-隐层 输出 — 隐层

输入 $x_1 \sim x_T$

隐层 $h_0 \sim h_T$ $h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$

输出 $y_1 \sim y_T$ $y_t = \sigma(W_{yh}h_t + b_y)$

总结

- RNN由输入、隐藏层、输出组成
- 输入是逐个输入
- 隐藏层--与当前输入、上一个隐藏层 有关，使用的激活函数为tanh
 - 使用tanh是因为在级联的时候，如果有一个输入值特别大，而没有tanh归一化[-1,1]的效果的话，连乘下去这个值到最后将会非常大，那么别的比较小的数就没有任何意义了
- 输出--只与当前隐藏层有关

1.3 RNN的优点、缺点

优点:	缺点:
模型大小不随输入长度的增加	计算缓慢
可以处理任意长序列	存在梯度消失和梯度爆炸的问题
考虑了时间的依赖	
参数随时间共享	

1.4 RNN中的梯度消失和爆炸

BPTT — 假设 $g(\cdot)$ 为链式变换, $h_t = W_{hx} x_t + W_{hh} h_{t-1} + b_h$

$$\Rightarrow \frac{\partial h_t}{\partial h_0} = \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial h_0} = W_{hh}^t$$

可对 W_{hh} 进行 SVD $\Rightarrow W_{hh} = \sum_{i=1}^r \sigma_i u_i v_i^T$

$$\therefore W_{hh}^t = \sum_{i=1}^r \sigma_i^t u_i v_i^T$$

$$\hookrightarrow \frac{\partial L}{\partial h_0} = \frac{\partial L}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_0} = W_{hh}^t \frac{\partial L}{\partial h_0} = \left(\sum_{i=1}^r \sigma_i^t u_i v_i^T \right) \frac{\partial L}{\partial h_0}$$

- 梯度爆炸：学习到的参数矩阵的最大奇异值大于1
 - 可以使用梯度裁剪解决(clipping)
- 梯度消失：学习到的参数矩阵的最大奇异值小于1
 - 当最大奇异值小于1的时候，往前传播的时候，随着t越来越大，梯度越来越小，越往前神经网络越不更新，削弱了RNN捕获长距离的能力

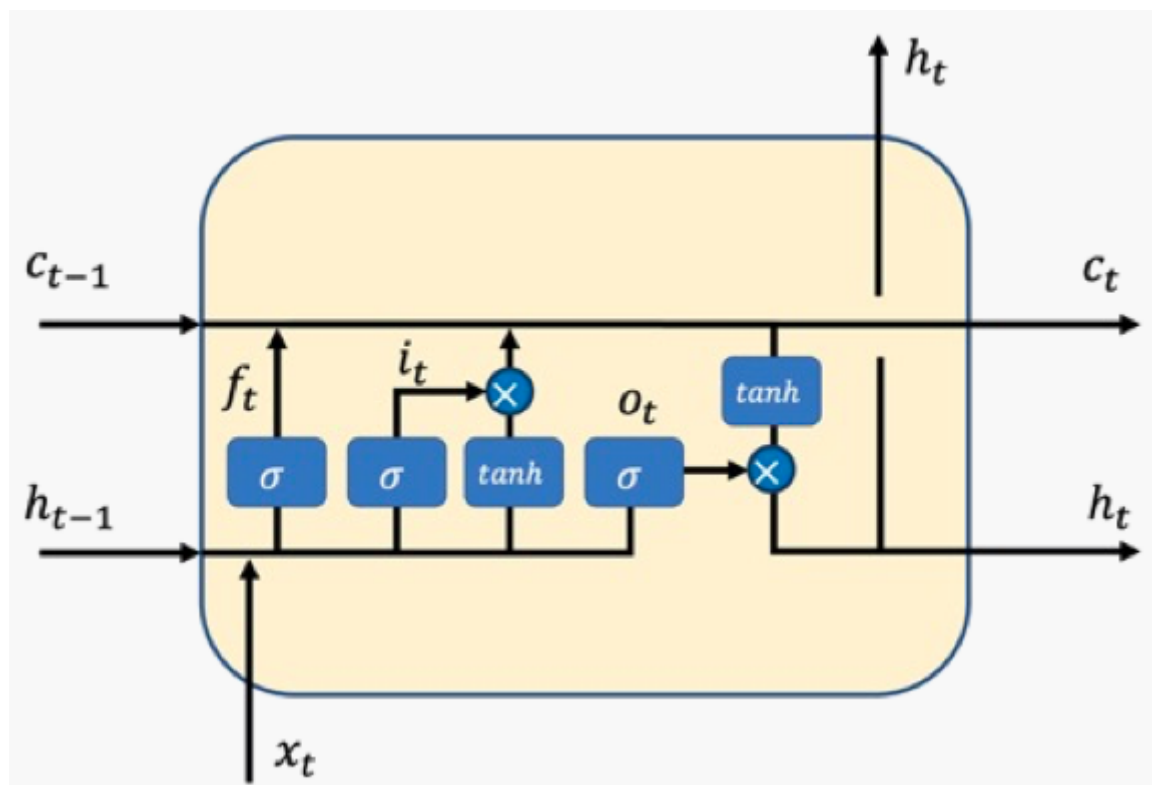
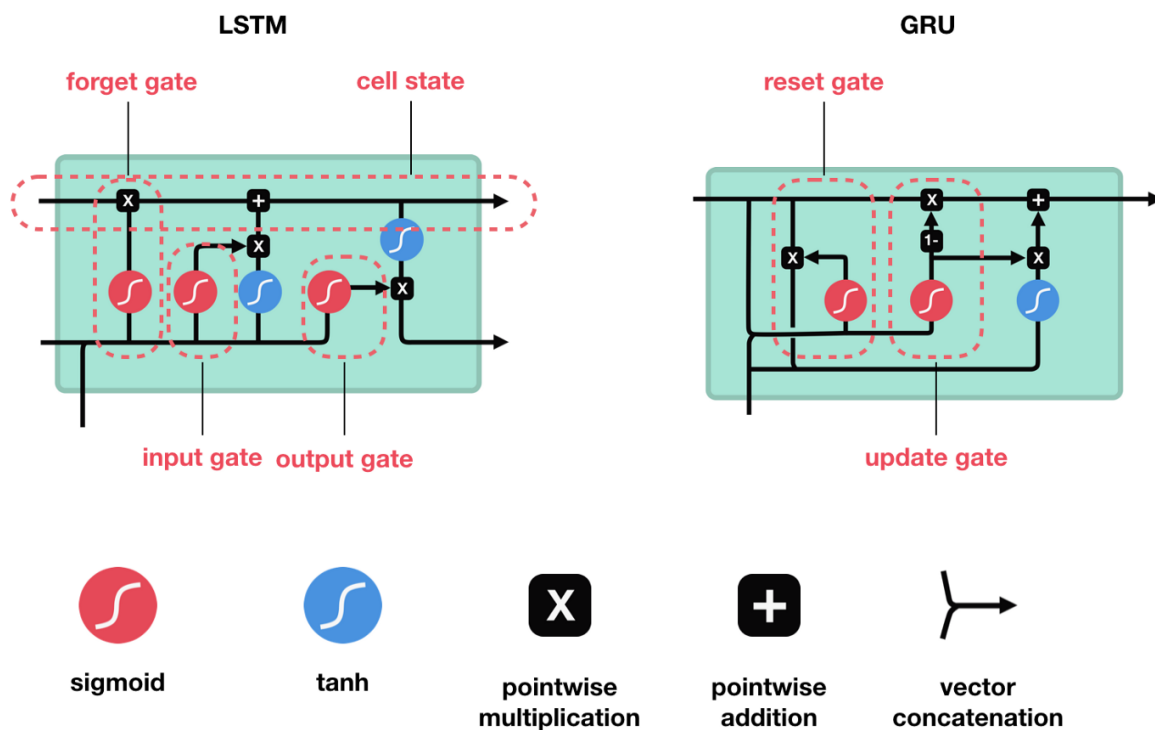
2.LSTM

2.1 LSTM的动机

有选择性的输入以缓解梯度消失的问题

2.2 LSTM的结构和公式

结构



公式--注意，公式中激活函数内的相加不是相加，而是向量拼接

公式：

3i) — sigmoid, 都与输入 x_t 和上一时刻/隐藏层有关 h_{t-1}

$$\text{输入 } i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1})$$

$$\text{遗忘 } f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1})$$

$$\text{输出 } o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1})$$

3态 —

暂态 \tilde{c}_t — x_t, h_{t-1} , 承担 RNN 中 h_t 的任务。有张

$$\hookrightarrow \tilde{c}_t = \tanh(W_{\tilde{c}x} x_t + W_{\tilde{c}h} h_{t-1})$$

细胞态 c_t — \tilde{c}_t, c_{t-1} , 调控。历史信息与当前信息。无张

$$\hookrightarrow c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1}$$

隐藏层 h_t — c_t , h_t 不是所有东西都与 c_t 有关。无张。

$$\hookrightarrow h_t = o_t \odot \tanh(c_t)$$

总结--3门3态

- 门：
 - 都与当前输入和上一时刻隐藏层相关
 - 都使用sigmoid作为激活函数
 - 都有可学习参数
- 输入门
 - 用于细胞态中调控暂态的输入
- 遗忘门
 - 用于细胞态中调控上一时刻细胞态的输入
- 输出门
 - 用于从细胞态生成当前时刻的隐藏层
- 态：

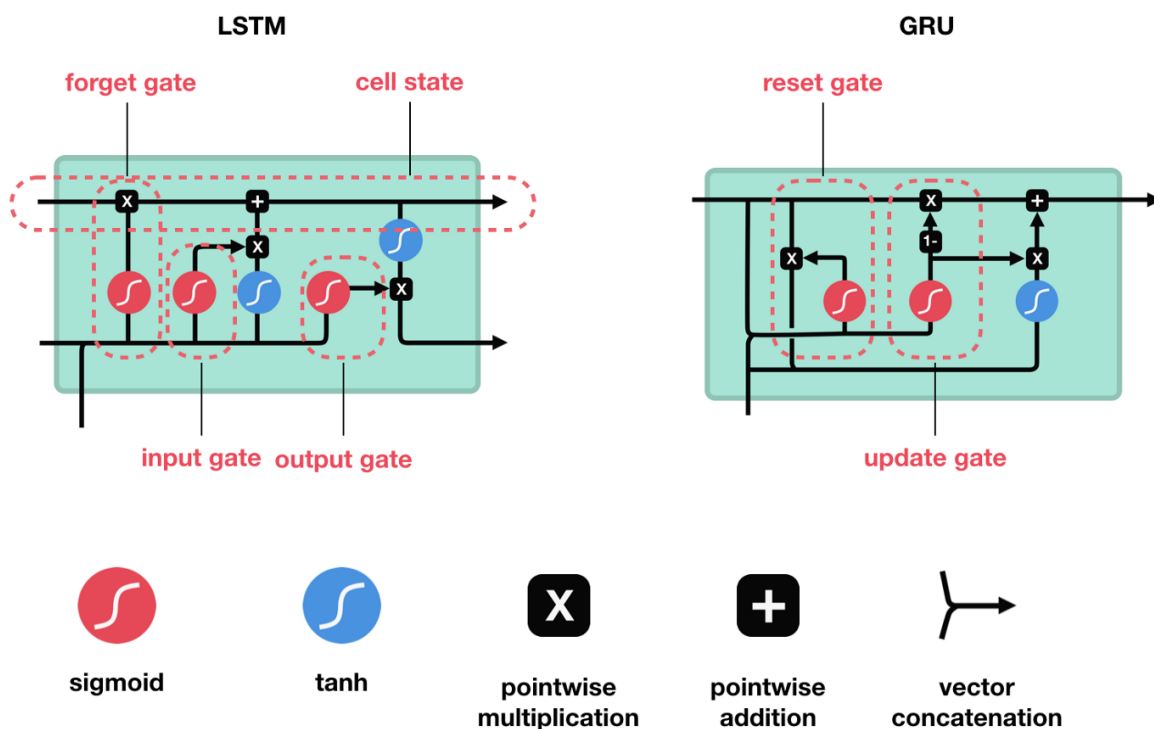
- 暂态
 - 与当前输入和上一时刻隐藏层相关
 - 使用tanh作为激活函数
 - 有可学习参数
- 细胞态
 - 由暂态与上一时刻细胞态决定
 - 由输入门和遗忘门调控输入
 - 无激活函数
 - 无可学习参数
- 隐藏层
 - 直接由细胞态经过tanh后经由输出门调控生成
 - 使用tanh作为激活函数
 - 无可学习参数

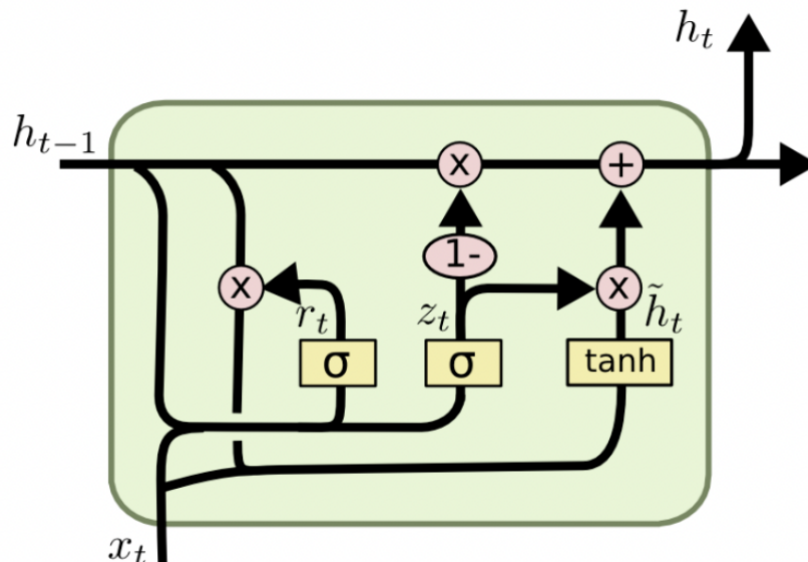
3.GRU

3.1 GRU的动机

3.2 GRU的结构和公式

结构





公式--注意，公式中激活函数内的相加不是相加，而是向量拼接

GRU

公式：

两门 —— sigmoid，都与当前输入和上一时刻隐藏层相关。

重置门 $r_t = \sigma(W_{rx} \cdot x_t + W_{rh} \cdot h_{t-1})$

更新门 $z_t = \sigma(W_{zx} \cdot x_t + W_{zh} \cdot h_{t-1})$

两态：

暂态 $\tilde{h}_t = \tanh(W_{hx} \cdot x_t + r_t \odot (W_{hz} \cdot h_{t-1}))$ (通过重置门控制过往)

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

(通过更新门控制更新)

总结--2门2态

- 门：
 - 都与当前输入和上一时刻的隐藏层有关，对于当前输入和上一时刻隐藏层都有可学习参数
 - 都使用sigmoid激活函数进行逐个元素相乘，相当于门控
- 重置门
 - 负责在暂态中遗忘上一时刻的隐藏层
- 更新门
 - 负责在隐藏层中调控暂态和上一时刻隐藏层的比例

- 态：
 - GRU只有暂态和隐藏层
- 暂态
 - 和RNN的隐藏层类似，只是 参数*上一时刻隐藏层 需要经 重置门 调控
 - tanh为激活函数
 - 有可学习参数
- 隐藏层
 - 由 暂态 和 上一时刻隐藏层 经由 更新门 调控输入比例
 - 无激活函数
 - 无可学习参数

4. 其他

4.0 比较

LSTM与GRU

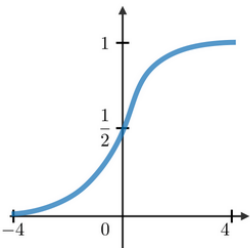
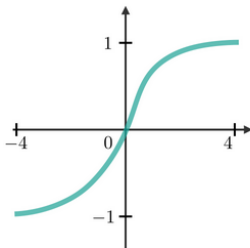
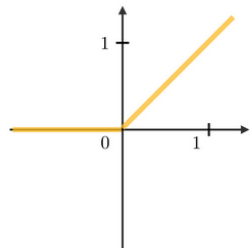
- LSTM能够解决循环神经网络因长期依赖带来的梯度消失和梯度爆炸问题，但是LSTM有三个不同的门，参数较多，训练起来比较困难。
- GRU只含有两个门控结构，且在超参数全部调优的情况下，二者性能相当，但是GRU结构更为简单，训练样本较少，易实现。
- GRU在LSTM的基础上主要做出了两点改变：

(1) GRU只有两个门。GRU将LSTM中的输入门和遗忘门合二为一，称为更新门（update gate），上图中的 z_t ，控制前边记忆信息能够继续保留到当前时刻的数据量，或者说决定有多少前一时间步的信息和当前时间步的信息要被继续传递到未来;GRU的另一个门称为重置门（reset gate），上图中的 r_t ，控制要遗忘多少过去的信息。

(2) 取消进行线性自更新的记忆单元（memory cell），而是直接在隐藏单元中利用门控直接进行线性自更新。GRU的逻辑图如上图所示。

4.1 常使用激活函数

□ **Commonly used activation functions** — The most common activation functions used in RNN modules are described below:

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

为何非门控单元的可学习参数，选择tanh而不也是sigmoid

- 在输入为0附近，tanh的梯度比sigmoid更大，学习收敛更快

可否使用RELU作为激活函数

- 使用RELU作为激活函数，则失去了tanh的约束，会引发梯度的消失和爆炸
- 如果可学习参数初始化在单位阵附近，则可能可以使用

4.2 参数量的计算

参考pytorch中的计算: <https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html?highlight=lstm#torch.nn.LSTM>

4.3 代码的书写

- [官方代码](#)

4.4 不同的整合方式:

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#overview>

❑ **Variants of RNNs** — The table below sums up the other commonly used RNN architectures:

