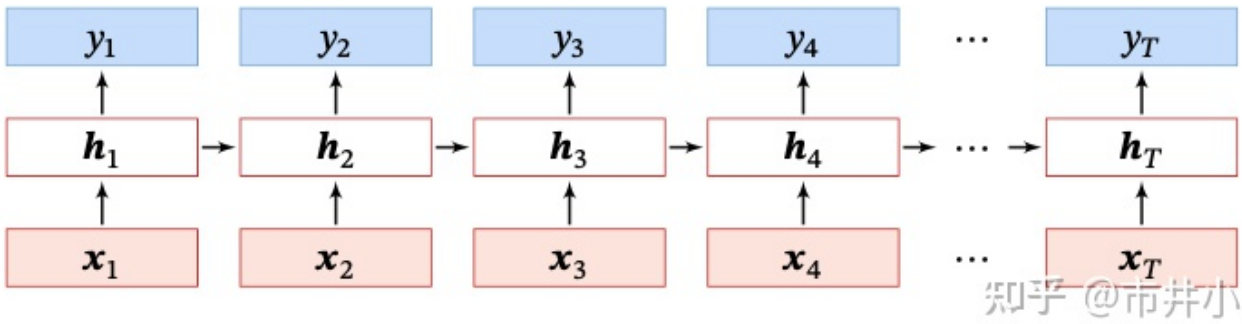
机器学习与深度学习面试系列十四（RNN）

循环神经网络(RNN)用来解决什么问题?

循环神经网络(Recurrent Neural Network，RNN)是用来建模序列化数据的一 种主流深度学习模型。

传统文本处理任务的方法中一般将TF-IDF向量作为特征输入，显然这样的表示实际上丢失了输入的文本序列中每个单词的顺序。在神经网络的建模过程中，一般的前馈神经网络，如卷积神经网络，通常接受一个定长的向量作为输入，无法处理变长的序列信息，即使通过滑动窗口加池化的方式将原先的输入转换成一个固定长度的向量表示，这样做可以捕捉到原文本中的一些局部特征，但是两个单词之间的长距离依赖关系还是很难被学习到。RNN则通过将神经元串行起来处 理序列化的数据。由于每个神经元能用它的内部变量保存之前输入的序列信息， 因此整个序列被浓缩成抽象的表示，并可以据此进行分类或生成新的序列。



RNN中反向传播（BPTT）是怎样的？为什么存在梯度消失问题?

BPTT 算法将循环神经网络看作是一个展开的多层前馈网络，其中“每一层” 对应循环网络中的“每个时刻”，如上图所示。对于t时刻，记 $\mathcal{L}_t$ 为t时刻的损失函数， z_t 和 h_t 分别为t时刻隐藏节点的输入值和输出值，则： $h_t = f(z_t)$ ，其中 f 为激活函数， $z_t = Uh_{t-1} + Wx_t + b$ ，也就是说t时刻隐藏节点的输入是由t-1时刻隐藏节点的输出和t时刻的新输入线性组合得到，我们要学习的参数就是 U 和 W 。上图只是我们按照时间维度将神经网络展开的一种形象化描述，实际上在神经网络在每个时刻的参数都是一样的，即 U 和 W 是共享的。

这样，循环神经网络就可以按照前馈网络中的反向传播算法计算参数梯度。在“展开”的前馈网络中，所有层的参数是共享的，因此参数的真实梯度是所有“展开层”的参数梯度之和。

$$\begin{aligned}\frac{\partial \mathcal{L}_t}{\partial U} &= \frac{\partial \mathcal{L}_t}{\partial z_t}\frac{\partial z_t}{\partial U} = \frac{\partial \mathcal{L}_t}{\partial z_t}\frac{\partial (Uh_{t-1}+Wx_t+b)}{\partial U} \\ &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+\frac{\partial z_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial U}) \\ &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+\frac{\partial z_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial z_{t-1}}\frac{\partial z_{t-1}}{\partial U}) \\ &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+U\frac{\partial h_{t-1}}{\partial z_{t-1}}\frac{\partial z_{t-1}}{\partial U})\end{aligned}$$

可以看出由于U在任意时刻都是共享的，$\frac{\partial z_t}{\partial U}$ 和 $\frac{\partial z_{t-1}}{\partial U}$ 存在递归关系，其中 $\frac{\partial h_{t-1}}{\partial z_{t-1}}$ 是激活函数的导数，简单记 $P_{t-1}=U\frac{\partial h_{t-1}}{\partial z_{t-1}}$ ：

$$\begin{aligned}\frac{\partial \mathcal{L}_t}{\partial U} &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+U\frac{\partial h_{t-1}}{\partial z_{t-1}}\frac{\partial z_{t-1}}{\partial U}) \\ &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+P_{t-1}\frac{\partial z_{t-1}}{\partial U}) \\ &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+P_{t-1}(h_{t-2}+P_{t-2}\frac{\partial z_{t-2}}{\partial U})) \\ &= \frac{\partial \mathcal{L}_t}{\partial z_t}(h_{t-1}+P_{t-1}h_{t-2}+P_{t-1}P_{t-2}h_{t-3}+\ldots+\prod_{i=2}^{t-1}P_i h_1)\end{aligned}$$

值得注意的一点是，这里的梯度消失/爆炸与我们在普通的前馈网络时说的不太一样。普通的前馈网络说梯度消失/爆炸时，梯度就真的逼近0或者无穷大了，但在循环神经网络中，我们看到对于 $\frac{\partial \mathcal{L}_t}{\partial U}$ 来说，他的括号内部是一个累加操作，可以认为是 t-1 个梯度的累加，此时可以看出一些端倪，当 P_i 的雅克比矩阵的最大特征值大于1时，随着离输出越来越远，每层的梯度大小 $\prod_{i=2}^{t-1}P_i$ 会呈指数增长，导致梯度爆炸。反之，若 P_i 的雅克比矩阵的最大特征值小于1，梯度的大小 $\prod_{i=2}^{t-1}P_i$ 会呈指数缩小， 产生梯度消失。

其实上，整个梯度不可能为0，因为连加操作中起码有 h_{t-1} ，但是越往前面的时刻追溯单项的梯度值越小。举极限的例子来说，如果最后一项 $\prod_{i=2}^{t-1}P_i h_1$ 极小，换句话说，t=1时刻的输出对t=100、t=1000这样距离较远的时刻的影响微乎其微，因此我们说，这样的循环神经网络难以捕捉长距离依赖关系，这就是循环神经网络的梯度消失问题。

为什么RNN激活函数通常选择tanh而不是ReLu?

通过上面的分析我们知道，RNN的梯度消失/爆炸来源于 P_{t-1} ，而 $P_{t-1}=\frac{\partial z_t}{\partial z_{t-1}}=\frac{\partial z_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial z_{t-1}}=U\frac{\partial h_{t-1}}{\partial z_{t-1}}$ ，如果激活函数是tanh， $P_{t-1}=U(1-h_{t-1}^2)$ ，虽然U的范围无法确定，但是如果U很大，则 $z_{t-1}=Uh_{t-2}+Wx_{t-1}+b$ 很大， $tanh(z_{t-1})$ 接近于1，导致 $(1-h_{t-1}^2)$ 很小。这样限制了 $P_{t-1}=U(1-h_{t-1}^2)$ 的大小不会太大，从而缓解了梯度爆炸的风险。

如果激活函数是ReLU，正半轴的梯度全为1， P_{t-1} 的大小完全取决于 U 的大小，梯度爆炸的风险更高。

如何缓解RNN梯度消失问题?

为了避免梯度爆炸或消失问题，一种最直接的方式就是选取合适的参数，同时使用非饱和的激活函数，尽量使得 $P_{t-1}=U\frac{\partial h_{t-1}}{\partial z_{t-1}}\approx 1$ ，但是这种方式需要足够的人工调参经验。所以更好的方法还是改进我们的模型。

从信息流传递的角度看， $h_t=f(Uh_{t-1}+Wx_t+b)$ ，正是因为这个激活函数和U的存在，导致信息传递的过程有损失，使得模型很难学习到距离较远的序列信息。如果能将信息流传递设计为 $h_t=h_{t-1}+g(x_t,\theta)$ ，这样就不存在信息传递过程中损失的问题了。更近一步，为了提高灵活性，我们可以将模型设计为 $h_t=h_{t-1}+g(x_t,h_{t-1},\theta)$ 。

但这样的设计还存在一个问题，就是记忆容量的问题。随着时间的增长， h_t 会变得越 来越大，而隐状态 h_t 可以存储的信息是有限的。所以LSTM和GRU设计了门控来进一步解决记忆容量的问题。

LSTM如何解决循环神经网络中的梯度消失问题?

LSTM 网络引入一个新的内部状态 c_t 专门进行线性的循环信息传递，同时(非线性地)输出信息给隐藏层的外部状态 h_t 。 内部状态 c_t 通过下面公式计算：

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

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

c_{t-1} 为上一时刻的记忆单元， $\hat{c}_t$ 是通过非线性函数得到的候选状态：

$$\hat{c}_t=tanh(U_ch_t+W_cx_t+b_c)$$

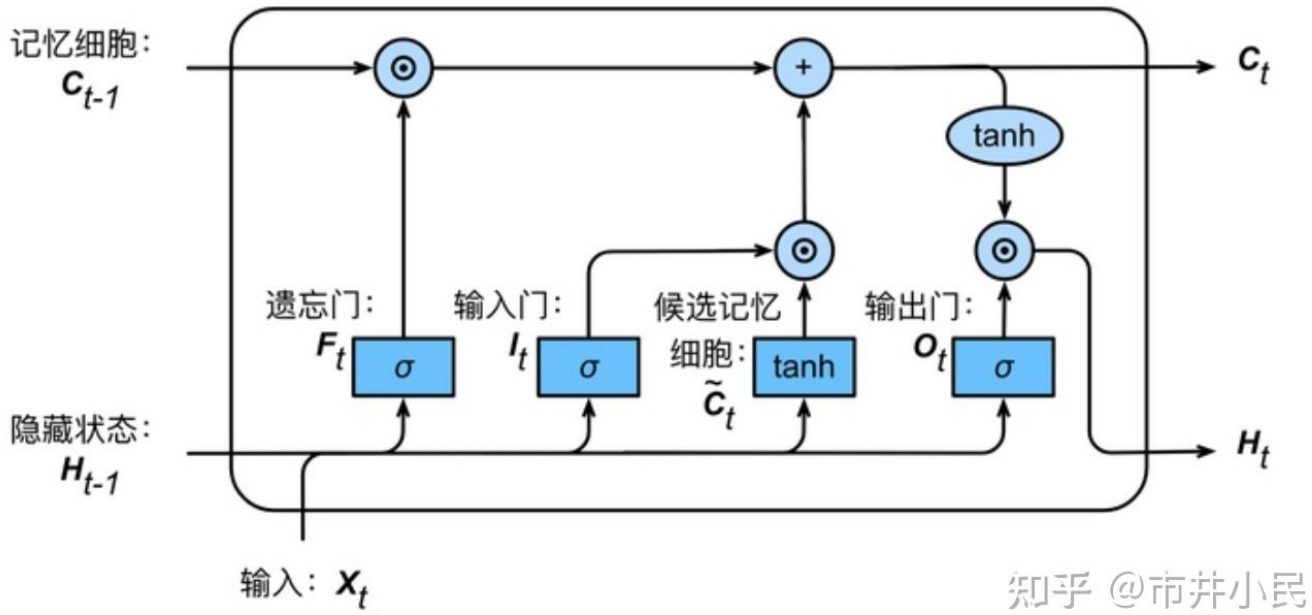
在每个时刻 t ，LSTM 网络的内部状态 c_t 记录了到当前时刻为止的历史信息。LSTM 网络引入门控机制来控制信息传递的路径， f_t,i_t,o_t 分别叫做遗忘门、输入门、输出门。

1. 遗忘门 f_t 控制上一个时刻的内部状态 c_{t-1} 需要遗忘多少信息。
2. 输入门 i_t 控制当前时刻的候选状态 $\hat{c}_t$ 有多少信息需要保存。
3. 输出门 o_t 控制当前时刻的内部状态 c_t 有多少信息需要输出给外部状态 h_t 。

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_i)$$



循环神经网络中的隐状态 h_t 存储了历史信息，可以看作是一种记忆。 在简单循环网络中，隐状态每个时刻都会被重写，因此可以看作是一种短期记忆(Short-Term Memory). 在LSTM神经网络中，长期记忆(Long-Term Memory)可以看作是网络参数，隐含了从训练数据中学到的经验，其更新周期要远远慢于短期记忆。记忆单元 c_t 可以在某个时刻捕捉到某个关键信息，并有能力将此关键信息保存一定的时间间隔。记忆单元 c_t 中保存信息的生命周期要长于短期记忆 h_t ，但又远远短于长期记忆，因此称为长短期记忆(Long Short-Term Memory)。遗忘的参数初始值一般都设得比较大，其偏置向量 b_f 设为 1 或 2，确保在开始训练时，LSTM网络能够捕捉到长距离的依赖信息。

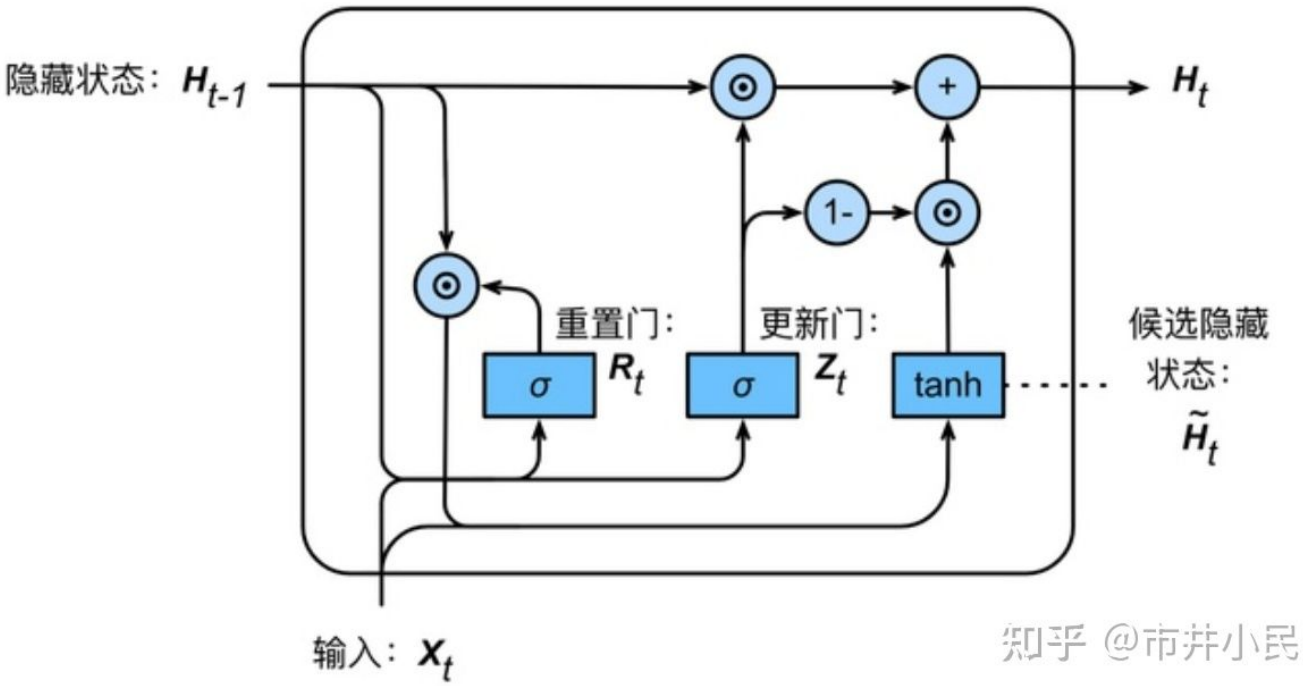
总的来说，LSTM 通过门控单元以及记忆单元的线性自循环，给梯度的长距离持续流通提供了路径，改变了之前循环神经网络中信息和梯度的传播方式，解决了长期依赖问题。

LSTM激活函数的选择?

遗忘门、输入门和输出门使用Sigmoid函数作为激活函数，因为输出在[0, 1]范围，天生符合“门”的概念。

在生成候选记忆和输出 h_t 时，使用双曲正切函数Tanh作为激活函数，因为Tanh函数在输入为0附近相比Sigmoid函数有更大的梯 度，通常使模型收敛更快。这里选择ReLU也可以，但ReLU更容易造成梯度爆炸，需要配合梯度裁剪。

GRU如何用两个门控单元来控制时间序列的记忆及遗忘行为?



和LSTM不同，GRU不引入额外的记忆单元 c_t ，而是像普通RNN一样，借助于 h_t 做信息流的传递。GRU只有两个门，更新门 z_t 和重置门 r_t 。

更新门可以理解为对LSTM中遗忘门和输入门的一种平衡：

$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot g(x_t, h_{t-1}, \theta)$ ，其中 $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$ 。

记 $\hat{h}_t = g(x_t, h_{t-1}, \theta) = tanh((W_h x_t + U_h(r_t \odot h_{t-1}) + b_h))$ ，其中重置门 $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$ ，可以看出重置门用来控制候选状态 $\hat{h}_t$ 的计算是否依赖上一时刻的状态 h_{t-1} 。

1. 当 $z_t = 0, r_t = 1$ 时，GRU网络退化为简单循环网络。
2. 当 $z_t = 0, r_t = 0$ 时，当前状态 h_t 只和当前输入 x_t 相关，和历史状态 h_{t-1} 无关。
3. 当 $z_t = 1$ 时，当前状态 $h_t = h_{t-1}$ 等于上一时刻状态 h_{t-1} ，和当前输入 x_t 无关。

LSTM和GRU的参数量比较?

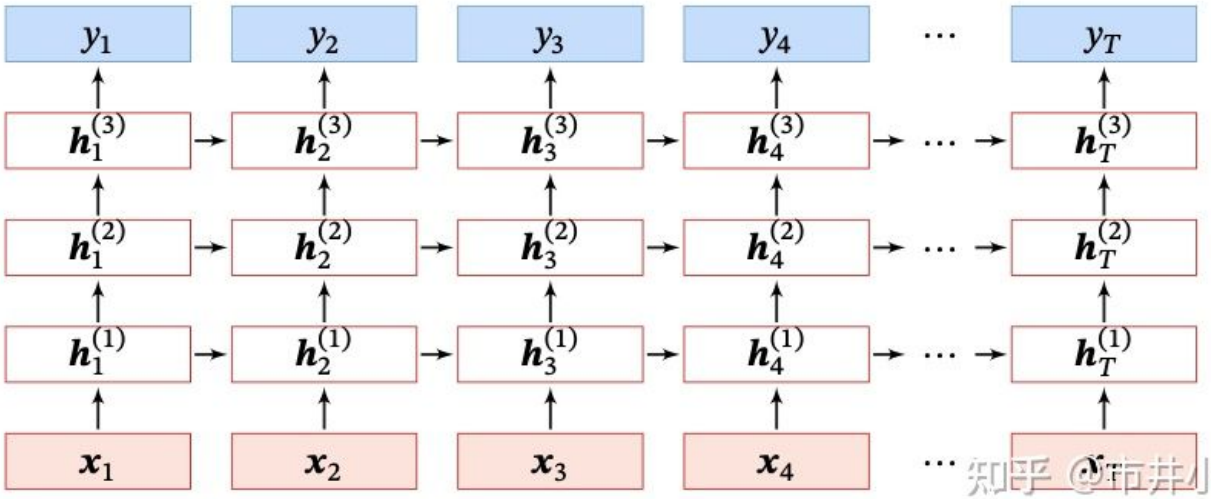
LSTM参数量大约是普通RNN的四倍， $W_i, U_i, b_i \quad i \in \{f, i, o, c\}$ ， f, i, o, c 分别代表三个门控和一个候选记忆细胞。

GRU参数量大约是普通RNN的三倍，$W_i, U_i, b_i \quad i \in \{z, r, h\}$ ， z, r, h 分别代表两个个门控和一个隐藏状态。

循环神经网络拓展深度有哪些方法?

循环神经网络可以看作是 既“深”又“浅”的网络。一方面来说，如果我们把循环网络按时间展开，长时间间隔的状态之间的路径很长，循环网络可以看作是一个非常深的网络。从另一方面 来说，如果同一时刻网络输入到输出之间的路径 $x_t \rightarrow y_t$ ，这个网络是非常浅的。

堆叠循环神经网络。就是像传统神经网络一样，直接将多个循环网络堆叠起来。



双向循环神经网络。在有些任务中，一个时刻的输出不但和过去时刻的信息有关，也和后续时刻的信息有关。比如给定一个句子，其中一个词的词性由它的上下文决定，即包含左右两边的信息。因此在这些任务中，我们可以增加一个按照时间的逆序来传递信息的网络层，来增强网络的能力。