


通俗理解RNN

2017年09月04日 11:14:42 水果先生 阅读数：14939

 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/qq_23225317/article/details/77834890

全连接神经网络和卷积神经网络他们都只能单独的取处理一个个的输入，前一个输入和后一个输入是完全没有关系的。但是，某些任务需要能够更好的处理序列的信息，即前面的输入和后面的输入是有关系的。比如，当我们在理解一句话意思时，孤立的理解这句话的每个词是不够的，我们需要处理这些词连接起来的整个序列；当我们处理视频的时候，我们也不能只单独的去分析每一帧，而要分析这些帧连接起来的整个序列。这时，就需要用到深度学习领域中另一类非常重要神经网络：循环神经网络(Recurrent Neural Network)。

RNN种类很多，也比较绕脑子。不过读者不用担心，本文将一如既往的对复杂的东西剥茧抽丝，帮助您理解RNNs以及它的训练算法，并动手实现一个循环神经网络。

语言模型

RNN是在自然语言处理领域中最先被用起来的，比如，RNN可以为语言模型来建模。那么，什么是语言模型呢？

我们可以和电脑玩一个游戏，我们写出一个句子前面的一些词，然后，让电脑帮我们写下接下来的一个词。比如下面这句：

1 我昨天上学迟到了，老师批评了____。
2

我们给电脑展示了这句话前面这些词，然后，让电脑写下接下来的一个词。在这个例子中，接下来的这个词最有可能是『我』，而不太可能是『小明』，甚至是『吃饭』。

语言模型就是这样的东西：给定一个一句话前面的部分，预测接下来最有可能的一个词是什么。

语言模型是对一种语言的特征进行建模，它有很多很多用处。比如在语音转文本(STT)的应用中，声学模型输出的结果，往往是若干个可能的候选词，这时候就需要语言模型来从这些候选词中选择一个最可能的。当然，它同样也可以用在图像到文本的识别中(OCR)。

使用RNN之前，语言模型主要是采用N-Gram。N可以是一个自然数，比如2或者3。它的含义是，假设一个词出现的概率只与前面N个词相关。我们以2-Gram为例。首先，对前面的一句话进行切词：

1 我 昨天 上学 迟到了，老师 批评了 ____。

如果用2-Gram进行建模，那么电脑在预测的时候，只会看到前面的『了』，然后，电脑会在语料库中，搜索『了』后面最可能的一个词。不管最后电脑选的是不是『我』，我们都知道这个模型是不靠谱的，因为『了』前面说了那么一大堆实际上是没有用到的。如果是3-Gram模型呢，会搜索『批评了』后面最可能的词，感觉上比2-Gram靠谱了不少，但还是远远不够的。因为这句话最关键的信息『我』，远在9个词之前！

现在读者可能会想，可以提升继续提升N的值呀，比如4-Gram、5-Gram.....。实际上，这个想法是没有实用性的。因为我们想处理任意长度的句子，N设为多少都不合适；另外，模型的大小和N的关系是指数级的，4-Gram模型就会占用海量的存储空间。

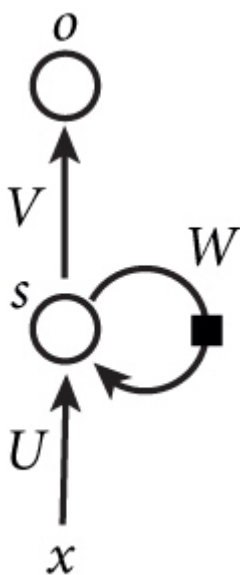
所以，该轮到RNN出场了，**RNN**理论上可以往前看(往后看)任意多个词。*单向*

循环神经网络是啥

循环神经网络种类繁多，我们先从最简单的基本循环神经网络开始吧。

基本循环神经网络

下图是一个简单的循环神经网络如，它由输入层、一个隐藏层和一个输出层组成：

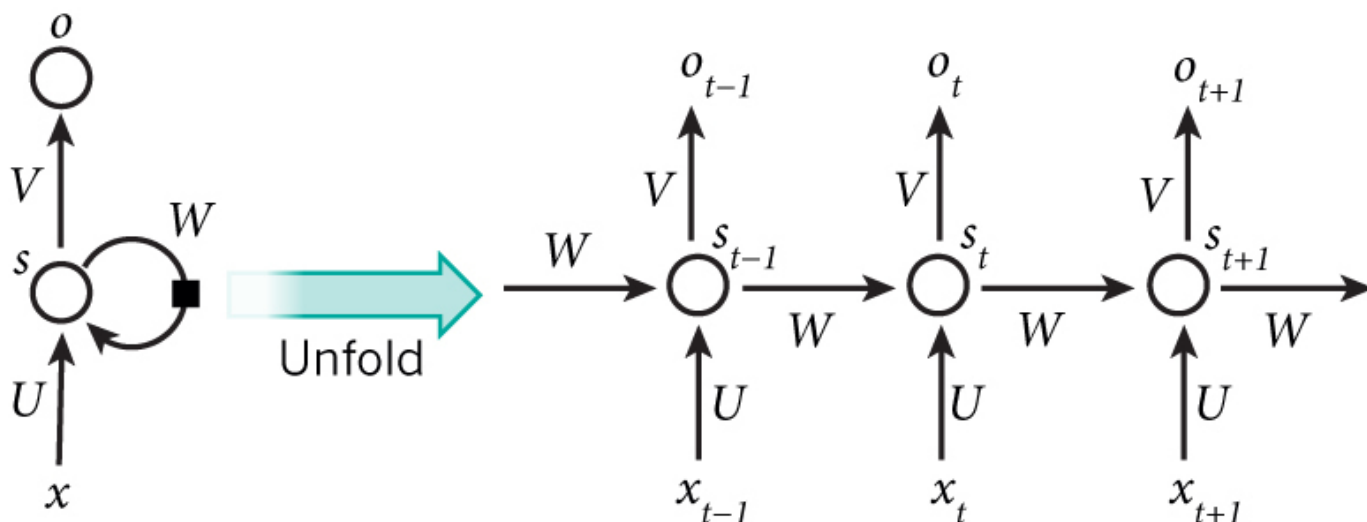


纳尼？！相信第一次看到这个玩意的读者内心和我一样是崩溃的。因为循环神经网络实在是太难画出来了，网上所有大神们都不得不用了这种抽象艺术手法。不过，静下心来仔细看看的话，其实也是很好理解的。

如果把上面有W的那个带箭头的圈去掉，它就变成了最普通的全连接神经网络。 x 是一个向量，它表示输入层的值（这里面没有画出来表示神经元节点的圆圈）； s 是一个向量，它表示隐藏层的值（这里隐藏层面画了一个节点，你也可以想象这一层其实是多个节点，节点数与向量 s 的维度相同）； U 是输入层到隐藏层的权重矩阵； o 也是一个向量，它表示输出层的值； V 是隐藏层到输出层的权重矩

阵。那么，现在来看看W是什么。循环神经网络的隐藏层的值s不仅仅取决于当前这次的输入x，还取决于上一次隐藏层的值s。权重矩阵W就是隐藏层上一次的值作为这一次的输入的权重。

如果我们将上面的图展开，循环神经网络也可以画成下面这个样子：



现在看上去就比较清楚了，这个网络在t时刻接收到输入 x_t 之后，隐藏层的值是 s_t ，输出值是 o_t 。关键一点是， s_t 的值不仅仅取决于 x_t ，还取决于 s_{t-1} 。我们可以用下面的公式来表示循环神经网络的计算方法：

$$o_t = g(Vs_t) \quad (1)$$

$$s_t = f(Ux_t + Ws_{t-1}) \quad (2)$$

式1是输出层的计算公式，输出层是一个全连接层，也就是它的每个节点都和隐藏层的每个节点相连。V是输出层的权重矩阵，g是激活函数。式2是隐藏层的计算公式，它是循环层。U是输入x的权重矩阵，W是上一次的值 s_{t-1} 作为这一次的输入的权重矩阵，f是激活函数。

从上面的公式我们可以看出，循环层和全连接层的区别就是循环层多了一个权重矩阵W。

如果反复把式2带入到式1，我们将得到：

$$o_t = g(Vs_t) \quad (3)$$

$$= Vf(Ux_t + Ws_{t-1}) \quad (4)$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2})) \quad (5)$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3}))) \quad (6)$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \dots)))) \quad (7)$$

从上面可以看出，循环神经网络的输出值 o_t ，是受前面历次输入值 x_t 、 x_{t-1} 、 x_{t-2} 、 x_{t-3} 、...影响的，这就是为什么循环神经网络可以往前看任意多个输入值的原因。

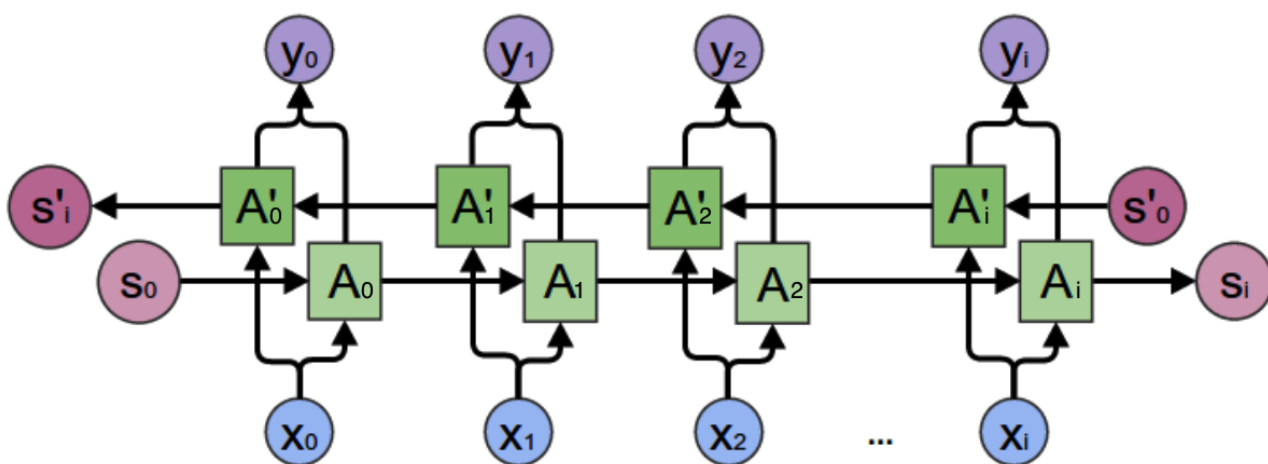
双向循环神经网络

对于语言模型来说，很多时候光看前面的词是不够的，比如下面这句话：

1 我的手机坏了，我打算_____一部新手机。

可以想象，如果我们只看横线前面的词，手机坏了，那么我是打算修一修？换一部新的？还是大哭一场？这些都是无法确定的。但如果我们也看到了横线后面的词是『一部新手机』，那么，横线上的词填『买』的概率就大得多了。

在上一小节中的基本循环神经网络是无法对此进行建模的，因此，我们需要双向循环神经网络，如下图所示：



当遇到这种从未来穿越回来的场景时，难免处于懵逼的状态。不过我们还是可以用屡试不爽的老办法：先分析一个特殊场景，然后再总结一般规律。我们先考虑上图中， y_2 的计算。

从上图可以看出，双向卷积神经网络的隐藏层要保存两个值，一个A参与正向计算，另一个值A'参与反向计算。最终的输出值 y_2 取决于 A_2 和 A'_2 。其计算方法为：

$$y_2 = g(VA_2 + V'A'_2)$$

A_2 和 A'_2 则分别计算：

$$A_2 = f(WA_1 + Ux_2) \quad (8)$$

$$A'_2 = f(W'A'_3 + U'x_2) \quad (9)$$

现在，我们已经可以看出一般的规律：正向计算时，隐藏层的值 s_t 与 s_{t-1} 有关；反向计算时，隐藏层的值 s'_t 与 s'_{t+1} 有关；最终的输出取决于正向和反向计算的加和。现在，我们仿照式1和式2，写出双向循环神经网络的计算方法：

$$o_t = g(Vs_t + V's'_t) \quad (10)$$

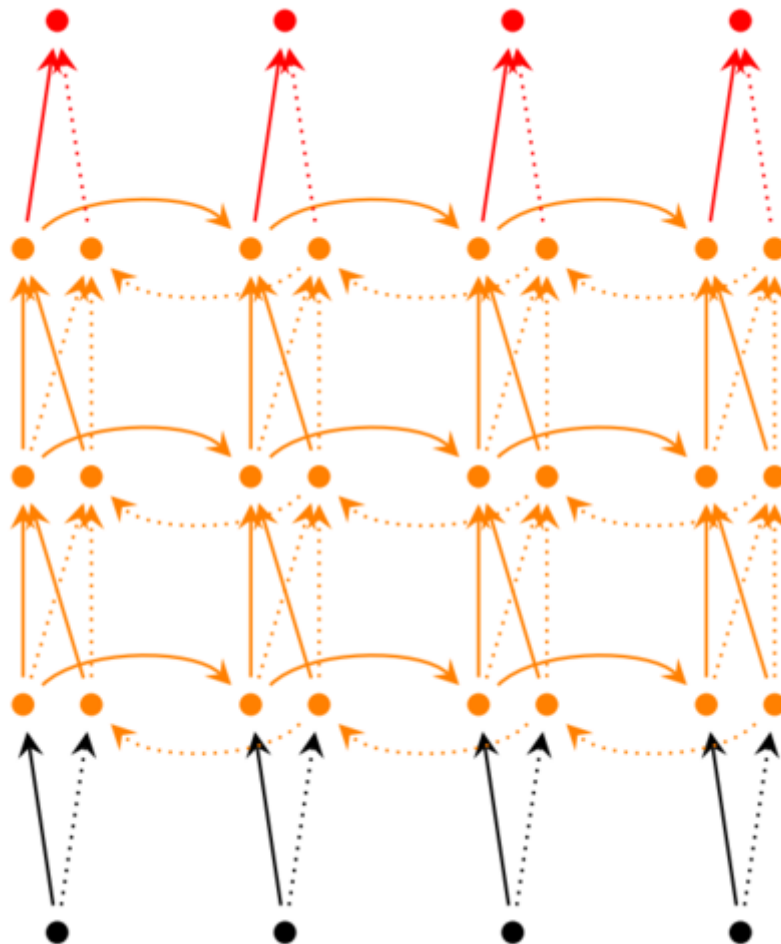
$$s_t = f(Ux_t + Ws_{t-1}) \quad (11)$$

$$s'_t = f(U'x_t + W's'_{t+1}) \quad (12)$$

从上面三个公式我们可以看到，正向计算和反向计算不共享权重，也就是说U和U'、W和W'、V和V'都是不同的权重矩阵。

深度循环神经网络

前面我们介绍的循环神经网络只有一个隐藏层，我们当然也可以堆叠两个以上的隐藏层，这样就得到了深度循环神经网络。如下图所示：



我们把第 i 个隐藏层的值表示为 $s_t^{(i)}$ 、 $s_t^{\prime(i)}$ ，则深度循环神经网络的计算方式可以表示为：

$$o_t = g(V^{(i)} s_t^{(i)} + V^{\prime(i)} s_t^{\prime(i)}) \quad (13)$$

$$s_t^{(i)} = f(U^{(i)} s_t^{(i-1)} + W^{(i)} s_{t-1}) \quad (14)$$

$$s_t^{\prime(i)} = f(U^{\prime(i)} s_t^{\prime(i-1)} + W^{\prime(i)} s_{t+1}^{\prime}) \quad (15)$$

$$\dots \quad (16)$$

$$s_t^{(1)} = f(U^{(1)} x_t + W^{(1)} s_{t-1}) \quad (17)$$

$$s_t^{\prime(1)} = f(U^{\prime(1)} x_t + W^{\prime(1)} s_{t+1}^{\prime}) \quad (18)$$

http://blog.csdn.net/qq_23225317

循环神经网络的训练

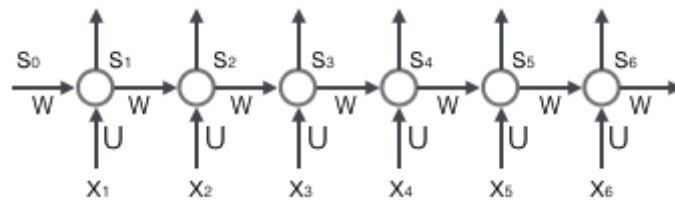
循环神经网络的训练算法：BPTT

BPTT算法是针对循环层的训练算法，它的基本原理和BP算法是一样的，也包含同样的三个步骤：

1. 前向计算每个神经元的输出值；
2. 反向计算每个神经元的误差项值，它是误差函数E对神经元j的加权输入的偏导数；
3. 计算每个权重的梯度。

最后再用随机梯度下降算法更新权重。

循环层如下图所示：



前向计算

使用前面的式2对循环层进行前向计算：