TensorFlow系列专题（九）：常用RNN网络结构及依赖优化问题

原创 磐创 磐创AI 2018-11-19

收录于话题
#TensorFlow专栏

27个

编辑 | 安可
出品 | 磐创AI技术团队

目录：

- **参考文献**

一、 常用的循环神经网络结构

前面的内容里我们介绍了循环神经网络的基本结构，这一小节里我们介绍几种更常用的循环神经网络的结构。

1. 多层循环神经网络

多层循环神经网络是最容易想到的一种变种结构，它的结构也很简单，就是在基本的循环神经网络的基础上增加了隐藏层的数量。

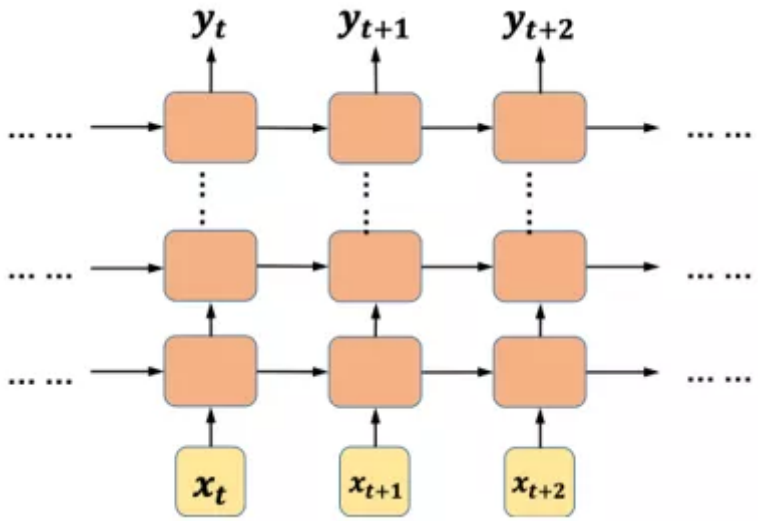

图1 多层循环神经网络结构

多层循环神经网络按时间展开后，每一层的参数和基本的循环神经网络结构一样，参数共享，而不同层的参数则一般不会共享（可以类比CNN的网络结构）。和基本结构的循环神经网络相比，多层循环神经网络的泛化能力更强，不过训练的时间复杂度和空间复杂度也更高。在TensorFlow里，我们可以借助MultiRNNCell这个类来实现深度循环神经网络，下面我们用一个具体的例子来演示TensorFlow中多层循环神经网络的实现。

示例：航班乘客人数预测

本示例所使用的数据来自DataMarket： https://datamarket.com/data/set/22u3/international-airline-passengers-monthly-totals-in-thousands-jan-49-dec-60#!ds=22u3&display=line。该数据集包含了从1949年到1960年共144个月的乘客总人数的数据，每个月的乘客人数是一条记录，共144条记录。其中部分数据显示如下：

时间	乘客人数（单位：千人）
"1949-01"	112
"1949-02"	118
"1949-03"	132
"1949-04"	129
"1949-05"	121

我们将所有数据可视化显示：

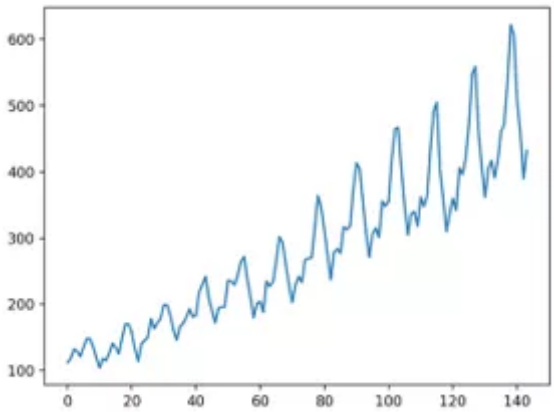

图2 数据可视化效果图

从可视化的效果可以看到数据的变化呈现周期性，对于RNN来说，学习这种很明显的序列关系并非难事。

我们定义一个多层循环神经网络（完整项目代码可以在本书配套的GitHub项目中找到）：

```
1  def get_cell(self):
2      return rnn.BasicRNNCell(self.hidden_dim)
3
4  def model(self):
5      # 定义一个多层循环神经网络
6      cells = rnn.MultiRNNCell([self.get_cell() for _ in range(3)], state_is_tuple=True)
7      # 用 cells 构造完整的循环神经网络
8      outputs, states = tf.nn.dynamic_rnn(cells, self.x, dtype=tf.float32)
9
10     num_examples = tf.shape(self.x)[0]
11     # 初始化输出层权重矩阵
12     W_out = tf.tile(tf.expand_dims(self.W_out, 0), [num_examples, 1, 1])
13     out = tf.matmul(outputs, W_out) + self.b_out    # 输出层的输出
14     out = tf.squeeze(out)    # 删除大小为 1 的维度
15     tf.add_to_collection('model', out)
16     return out
```

在第1行代码中定义了一个方法用来返回单层的cell，在第6行代码中，我们使用MultiRNNCell类生成了一个3层的循环神经网络。最终的预测结果如下图左侧所示：

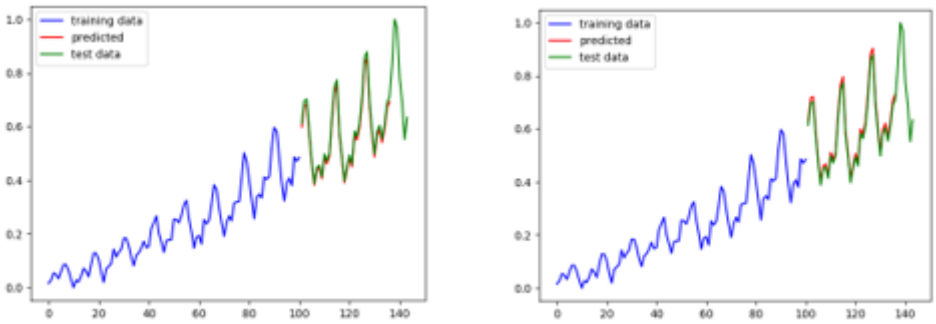

图3 多层循环神经网络的预测结果

上图中，蓝色线条是原始数据，绿色线条是从原始数据中划分出来的测试数据，红色线条是在测试数据上的预测效果，左侧是多层循环神经网络的预测结果，右侧是单层循环神经网络的预测结果（除网络层数不同外，其它参数均相同）。可以看到两个预测结果都几乎和真实数据重合，仔细比较会发现，多层循环神经网络的拟合效果更好一些。

2. 双向循环神经网络

无论是简单循环神经网络还是深度循环神经网络[1]，网络中的状态都是随着时间向后传播的，然而现实中的许多问题，并不都是这种单向的时序关系。例如在做词性标注的时候，我们需要结合这个词前后相邻的几个词才能对该词的词性做出判断，这种情况就需要双向循环神经网络来解决问题。

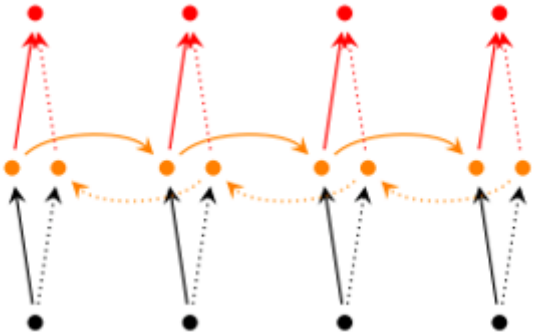

图4 双向循环神经网络结构

图片来源于http://www.wildml.com

双向循环神经网络可以简单的看成是两个单向循环神经网络的叠加，按时间展开后，一个是从左到右，一个是从右到左。双向循环神经网络的计算与单向的循环神经网络类似，只是每个时刻的输出由上下两个循环神经网络的输出共同决定。双向循环神经网络也可以在深度上进行叠加：

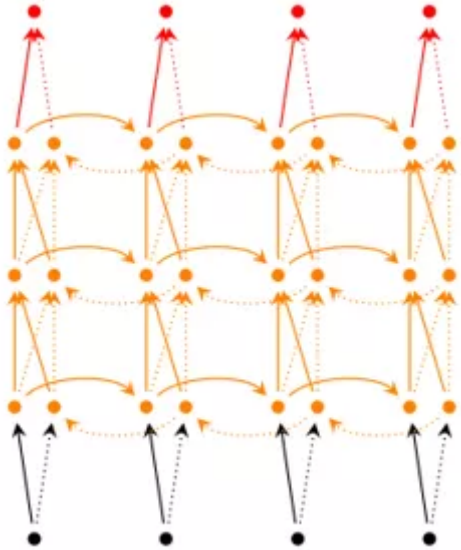

图5 深度双向循环神经网络结构

图片来源于http://www.wildml.com

在下一章的项目实战部分，我们会使用TensorFlow来实现深度双向循环神经网络解决文本多分类问题，会结合代码来介绍双向循环神经网络。

3. 递归神经网络

递归神经网络[2]（recursive neuralnetwork，RNN）是循环神经网络的又一个变种结构，看它们的名称缩写，很容易将两者混淆（通常我们说的RNN均特指recurrent neural network，同时也不建议将recurrent neural network说成是递归神经网络。一般默认递归神经网络指的是recursive neural network，而循环神经网络指的是recurrentneural network）。我们前面所介绍的循环神经网络是时间上的递归神经网络，而这里所说的递归神经网络是结构上的递归。递归神经网络相较于循环神经网络有它一定的优势，当然这个优势只适用于某些特定场合。目前递归神经网络在自然语言处理和计算机视觉领域已经有所应用，并获得了一定的成功，但是由于其对输入数据的要求相对循环神经网络更为苛刻，因此并没有被广泛的应用。

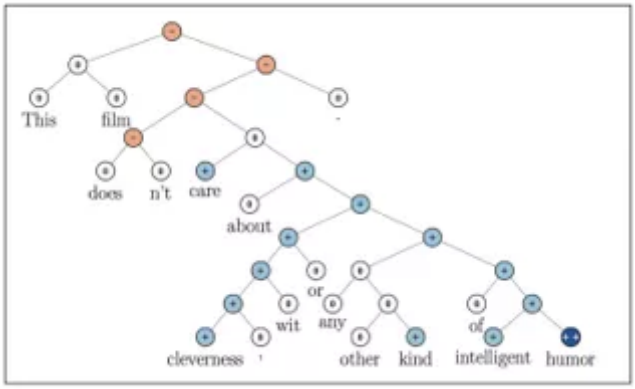

图6 递归神经网络结构示意图

二、 长期依赖问题及其优化

1. 长期依赖问题

什么是长期依赖？我们知道循环神经网络具有记忆能力，凭着它的记忆能力，能够较好的解决一般的序列问题，这些序列问题的数据内部基本上都存在着一定的依赖性，例如我们在前面提到过的词性标注的问题以及我们在演示项目里面人工构造的二进制数据。在有些现实问题中，数据间的依赖都是局部的、较短时间间隔的依赖。还是以词性标注为例，判断一个词是动词还是名词，或者是形容词之类，我们往往只需要看一下这个词前后的两个或多个词就可以做出判断，这种依赖关系在时间上的跨度很小。

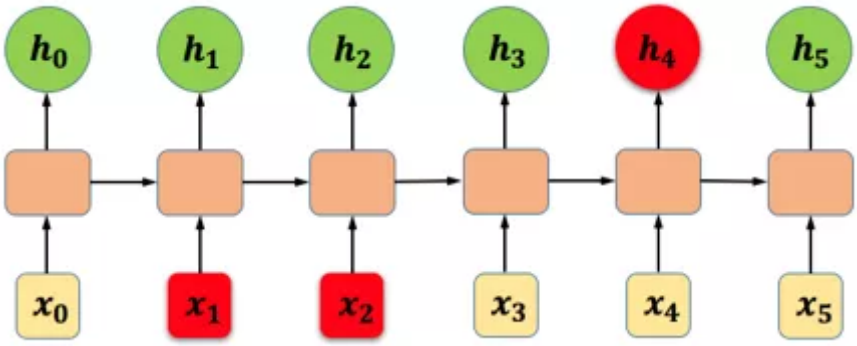

图7 时间跨度较小的依赖关系示意图

如上图所示，时刻网络的输出（上图中到是隐藏层的输出）除了与当前时刻的输入相关之外，还受到和时刻网络的状态影响。像这种依赖关系在时间上的跨度较小的情况下，RNN基本可以较好地解决，但如果出现了像图6-12所示的依赖情况，就会出现长期依赖问题：梯度消失和梯度爆炸。

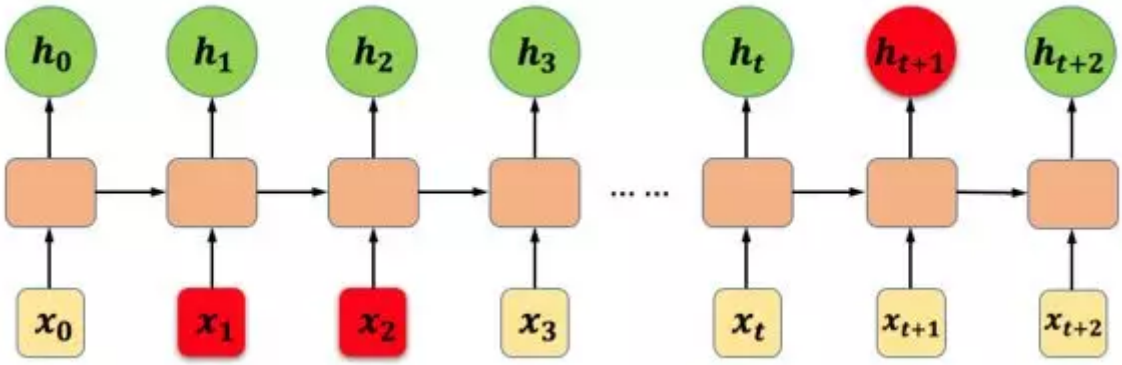

图8 时间跨度较大的依赖关系示意图

什么是梯度消失和梯度爆炸？图9是一个较为形象的描述，在深层的神经网络中，由于多个权重矩阵的相乘，会出现很多如图所示的陡峭区域，当然也有可能会出现很多非常平坦的区域。在这些陡峭的地方，Loss函数的倒数非常大，导致最终的梯度也很大，对参数进行更新后可能会导致参数的取值超出有效的取值范围，这种情况称之为梯度爆炸。而在那些非常平坦的地方，Loss的变化很小，这个时候梯度的值也会

很小（可能趋近于0），导致参数的更新非常缓慢，甚至更新的方向都不明确，这种情况称之为梯度消失。

长期依赖问题的存在会导致循环神经网络没有办法学习到时间跨度较长的依赖关系。

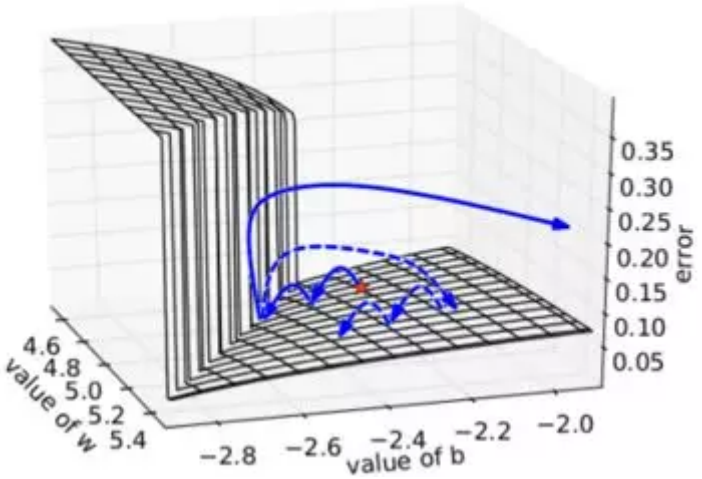

图9 导致梯度爆炸的情况

图片引用自Razvan Pascanu的论文：《On thedifficulty of training recurrent neural networks》

正如上面所说，长期依赖问题普遍存在于层数较深的神经网络之中，不仅存在于循环神经网络中，在深层的前馈神经网络中也存在这一问题，但由于循环神经网络中循环结构的存在，导致这一问题尤为突出，而在一般的前馈神经网络中，这一问题其实并不严重。

值得注意的是，在第三章中介绍激活函数的时候我们其实已经提到过梯度消失的问题，这是由于Sigmoid型的函数在其函数图像两端的倒数趋近于0，使得在使用BP算法进行参数更新的时候会出现梯度趋近于0的情况。针对这种情况导致的梯度消失的问题，一种有效的方法是使用ReLU激活函数。但是由于本节所介绍的梯度消失的问题并不是由激活函数引起的，因此使用ReLU激活函数也无法解决问题。下面我们来看一个简单的例子。

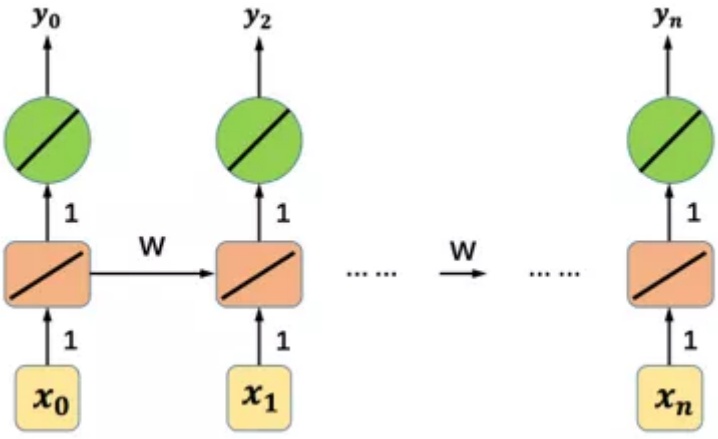

图10 参数W在循环神经网络中随时间传递的示意图

如上图所示，我们定义一个简化的循环神经网络，该网络中的所有激活函数均为线性的，除了在每个时间步上共享的参数W以外，其它的权重矩阵均设为1，偏置项均设为0。假设输入的序列中除了的值为1以外，其它输入的值均为0，则根据前一篇讲解RNN的原理内容，可以知道：

$$
\begin{aligned}
h_0 &= 1 \\
h_1 &= W \\
h_2 &= W^2 \\
&\vdots \\
h_n &= W^n
\end{aligned}
$$

公式1

最终可以得到，神经网络的输出是关于权重矩阵W的指数函数。当W的值大于1时，随着n的增加，神经网络最终输出的值也成指数级增长，而当W的值小于1时，随着n的值增加，神经网络最终的输出则会非常小。这两种结果分别是导致梯度爆炸和梯度消失的根本原因。

从上面的例子可以看到，循环神经网络中梯度消失和梯度爆炸问题产生的根本原因，是由于参数共享导致的。

2. 长期依赖问题的优化

对于梯度爆炸的问题，一般来说比较容易解决。我们可以用一个比较简单的方法叫做“梯度截断”，“梯度截断”的思路是设定一个阈值，当求得的梯度大于这个阈值的时候，就使用某种方法来进行干涉，从而减小梯度的变化。还有一种方法是给相关的参数添加正则化项，使得梯度处在一个较小的变化范围内。

梯度消失是循环神经网络面临的最大问题，相较于梯度爆炸问题要更难解决。目前最有效的方法就是在模型上做一些改变，这就是我们下一节要介绍的门控循环神经网络。本篇主要介绍了RNN的一些常用的循环网络结构，以及长期问题和如何优化。下一篇我们将介绍基于门控制的循环神经网络，也就是大家常见的LSTM以及GRU模型原理。

参考文献[1]BiDirectional_Recurrent_Neural_Networks_For_Speech_Recognition,MikeSchuster.

[2]RecursiveDeep Models for Semantic Compositionality Over a Sentiment Treebank