**PyTorch实现循环神经网络一-RNN的BPTT和梯度消失**

上次文章我们学习了循环神经网络的基础知识，具体文章请查看公众号历史文章。本次我们来聊聊循环神经网络（RNN）的梯度消失的问题。

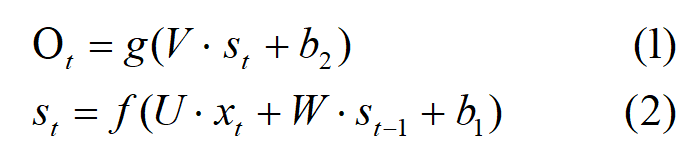
1. **BPTT算法**

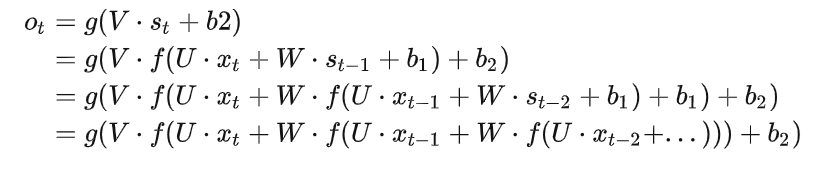
首先，要明白的一个道理是，BPTT并不是一个新的算法，BPTT只是标准反向传播算法（BP算法）在展开的循环神经网络上应用。为了区分一下与原BP算法的不同而取的。

我们已经知道，RNN网络基于实践中展开的结果如下所示：



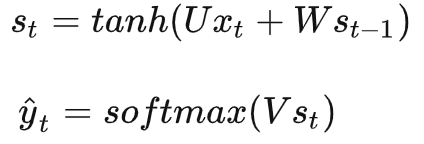
且对于t时刻，RNN的输出可以由以下公式描述：



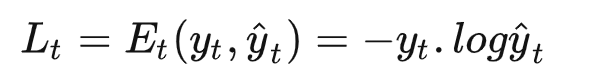


上述过程即RNN的前馈计算过程，现在我们来看看RNN的反向传播算法BPTT（Backpropagation Through Time），可以翻译为基于时间的反向传播。

我们具象化上述激活函数（使用具体的激活函数，且忽略偏置项），则有：



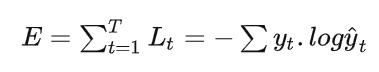
损失函数的定义为交叉熵损失（cross entrophy loss）:



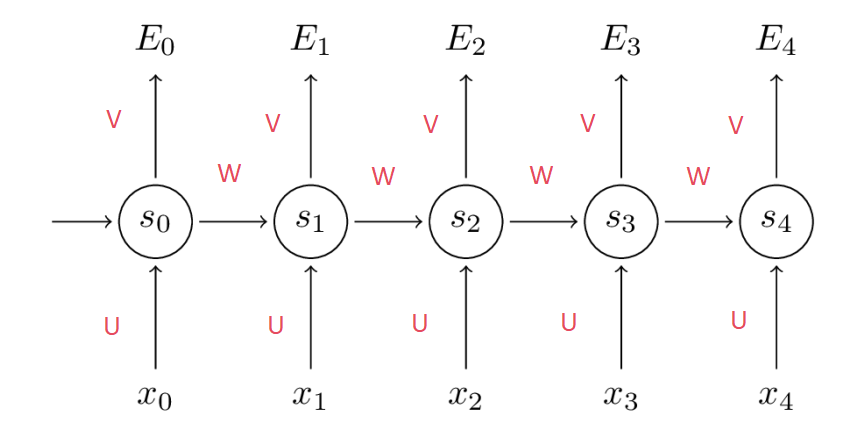
那么对于一个输入序列为：



的数据，其整体的损失可以记为：

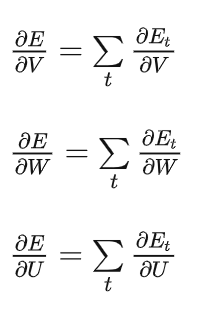


接下来我们对权重矩阵U，W，V进行求导，然后使用反向传播算法来进行参数的更新。这里我们参考[Denny](https://dennybritz.com/)大牛的博客，简化一个RNN结构图：

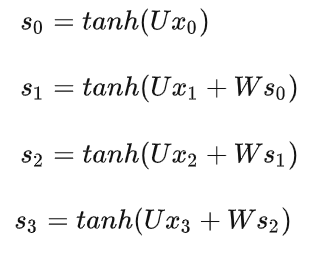


其中，E0,E1,...,E4为各个时刻的损失，假设总损失E为他们的和。

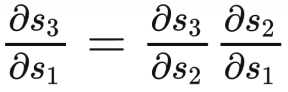
总误差E对各个参数矩阵的求导等于各个误差求导的总和，即：



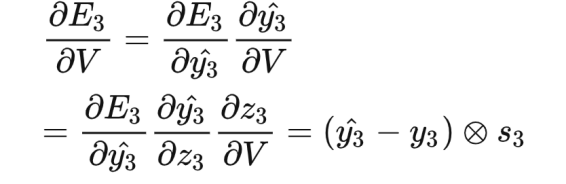
根据上述简图我们有：



可以看出，不同时刻的S具有依赖关系，比如说S3依赖S2,S2依赖S1。也即存在：



具体地，当t=3的时候，我们进行单项损失的求导，根据链式法有：



其中：



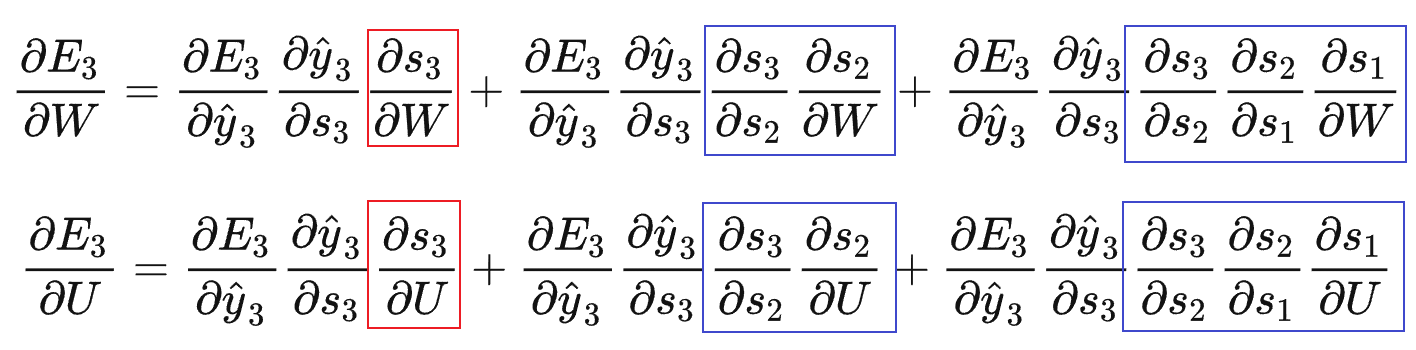
表示代表的是计算两个向量的外积，关于t=3的时候对V的求导我们就完成了，从求导的结果可以看出导数结果与哪些变量有关。

因为S状态之间的依赖关系：



比如对S3关于U进行求导的时候，还要考虑S2关于U的导数以及S1关于U的导数。这一点大家仔细的品味。

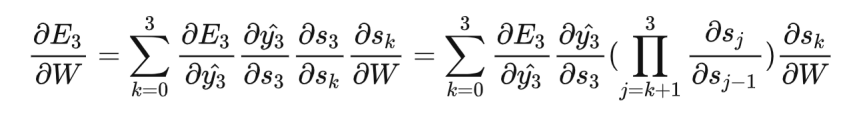
我们得到：



详细过程可以参考[Denny](https://dennybritz.com/)大牛的博客图片（感谢大佬的文章）：



进一步的，我们进一步来简化计算结果，则有：



没有什么太大的感情，全是技巧。

对上述公式的一个直观的解释就是，W在t0,t1,t2,t3时刻都对E3产生一定的影响，因此我们在计算梯度损失的时候，需要将梯度从时刻t=3通过网络反向传播到t=0时刻。

这样我们就完成了权重梯度的计算，这里不在详细推到误差项的计算了，感兴趣的小伙伴可以参考以下链接进行学习：

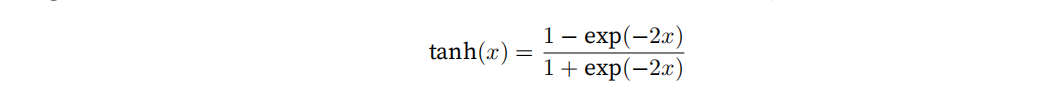
<https://zybuluo.com/hanbingtao/note/541458>

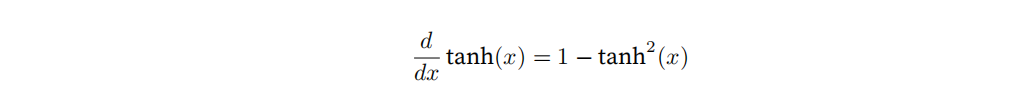
1. **RNN网络的梯度消失和爆炸**

按照之前的描述，我们有：

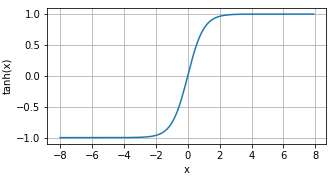


且tanh函数以及其导函数形式如下：

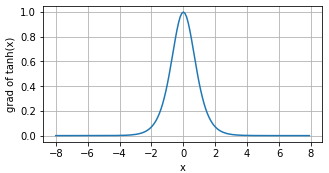
****



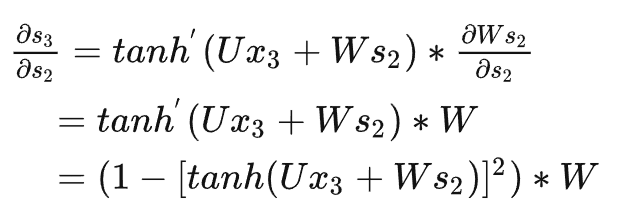
激活函数tanh的函数图像如下：



以及其导函数的图像如下：



这里计算一个隐藏层结果与上一个隐藏层之间的导数（梯度在隐藏层之间变换）：



由于tanh的导函数的值在(0,1]之间，对W的讨论为：

1. 如果W初始化的很小，则上述梯度可能会变得很小，就存在梯度消失的问题。
2. 如果W初始化的很大，则上述梯度可能会变得很大，就存在梯度爆炸的问题

大家可以从级数的敛散性去考虑这个问题。

我们在基于BPTT求梯度，必须在内存中保存各个时刻的RNN层的中间数据。因此，随着时序数据变长，计算机的内存使用量将会变得越来越大。**另外，梯度消失并不是说整个RNN网络的梯度都消失了，而是在靠近输出层的部分梯度较大，越往前梯度越小，时间开始的越早梯度消失的概率更大。**

对于RNN的梯度消失和爆炸的问题的解决方案，很多文章介绍了梯度截断的相关处理方法，实际上呢我们可以有更好的处理方法，如长短时记忆网络（LTSM）和Gated Recurrent Unit（GRU）。

文章之中，有什么错误之处，还请大家帮忙指出。

1. **参考文献**
2. <https://zybuluo.com/hanbingtao/note/541458>
3. <https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-3/>
4. http://proceedings.mlr.press/v28/pascanu13.pdf

1、[Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs](http://link.zhihu.com/?target=http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/" \t "https://zhuanlan.zhihu.com/p/_blank)

2、[Recurrent Neural Networks Tutorial, Part 2 – Implementing a RNN with Python, Numpy and Theano](http://link.zhihu.com/?target=http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/" \t "https://zhuanlan.zhihu.com/p/_blank)

3、[Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients](http://link.zhihu.com/?target=http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/" \t "https://zhuanlan.zhihu.com/p/_blank)

4、[Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano](http://link.zhihu.com/?target=http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/" \t "https://zhuanlan.zhihu.com/p/_blank)

https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-3/

http://colah.github.io/posts/2015-08-Understanding-LSTMs/