**PyTorch实现循环神经网络一-RNN基础学习**

这次的文章我们开始接触一下大名鼎鼎的循环神经网络，循环神经网络（RNN,Recurrent Neural Network）的文章网络上已经有很多了，那么作者为什么还要自己写呢？主要原因是想从一个小白的角度和大家分享一下循环神经网络的相关知识，在写作的过程中还可以发现一些不一样的东西和巩固自己已经掌握的知识点。

1. **为什么需要RNN**

RNN（Recurrent Neural Network）中的Recurrent是“反复发生、重复发生、周期性地发生、循环”，因此RNN可以直译为“复发神经网络”或者“循环神经网络”，这里使用后者。

这里的循环指的是经过一定时间又回到这个地点，然后重复进行。当然，这里的循环遵循“环路”的原则。在循环的过程中，数据才能在相同的地点之间来回移动，信息不断被更新。

那么为什么需要RNN呢，以自然语言中一个较为典型的任务（命名实体识别）进行分析：

第一句表述：I like eating apple very much.

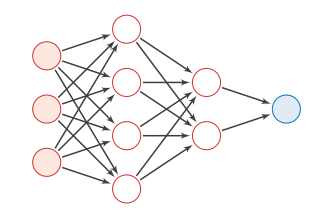
第二句表述：The Apple is a company that sells products such as the iPhone.

显然，在第一句话中，apple是指实物苹果，在第二句话中apple指的是指苹果公司。现在这两者是不一样的，基于前馈神经网络在没有考虑词语”apple”上下文的词的关系，因此基于前馈神经网络我们无法去完成这样的任务，因此需要具有记忆单元的神经网络，在学习输入语句的时候记住上下文词语之间的关系信息，这样就可以完成实体识别等任务。

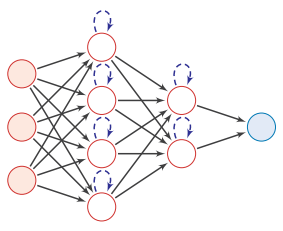
另外，我们熟知的时间序列预测问题中，由于这些序列型的数据往往都是具有时序上的关联性的，也就是说网络的输出除了与当前的输入相关之外，还与之前某一时刻或某几个时刻的输出相关。因此需要与记忆能力的网络，才能将前面的信息传递到后面。

1. **RNN的结构**

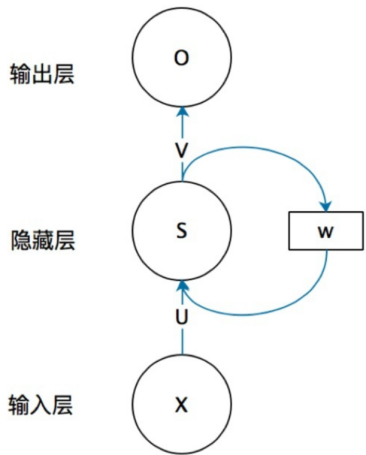
回想到之前我们学过的前馈神经网络，它的结构是这样的：



具有记忆功能的网络的结构是这样的（其中中间两层为隐藏层）：



具体而言，RNN的结构如所示：

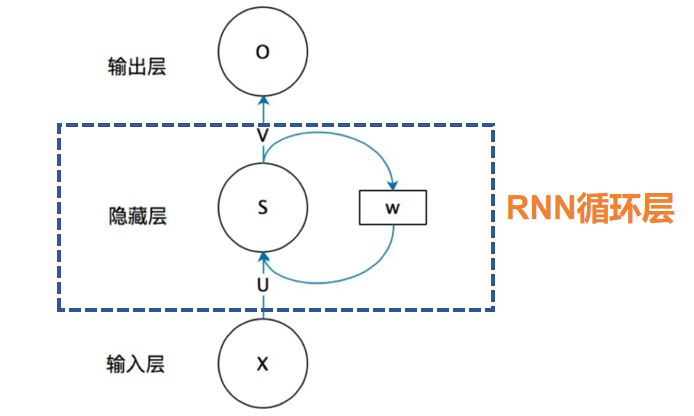


其中：

1. O为一个向量，为输出层的向量值
2. V为隐藏层到输出层的参数（矩阵）
3. S为隐藏层的向量
4. U为输入层到隐藏层的参数（矩阵）
5. X为一个向量，为输入层的向量值
6. W为隐藏层中上一层到下一层的权重矩阵

很显然，如果我们将上述图片中W去掉（包含箭头）部分，此时网络结构就变成了普通的全连接神经网络了。

在一些参考文献中，会将中间的隐藏层和“循环”称为RNN层，即：



接下来我们基于时间线将RNN循环层进行展开（在t时刻）：



通过展开图我们可以发现，t时刻隐藏层的输出不仅与t时刻的输入有关，还与t-1时刻隐藏层的输出有关。

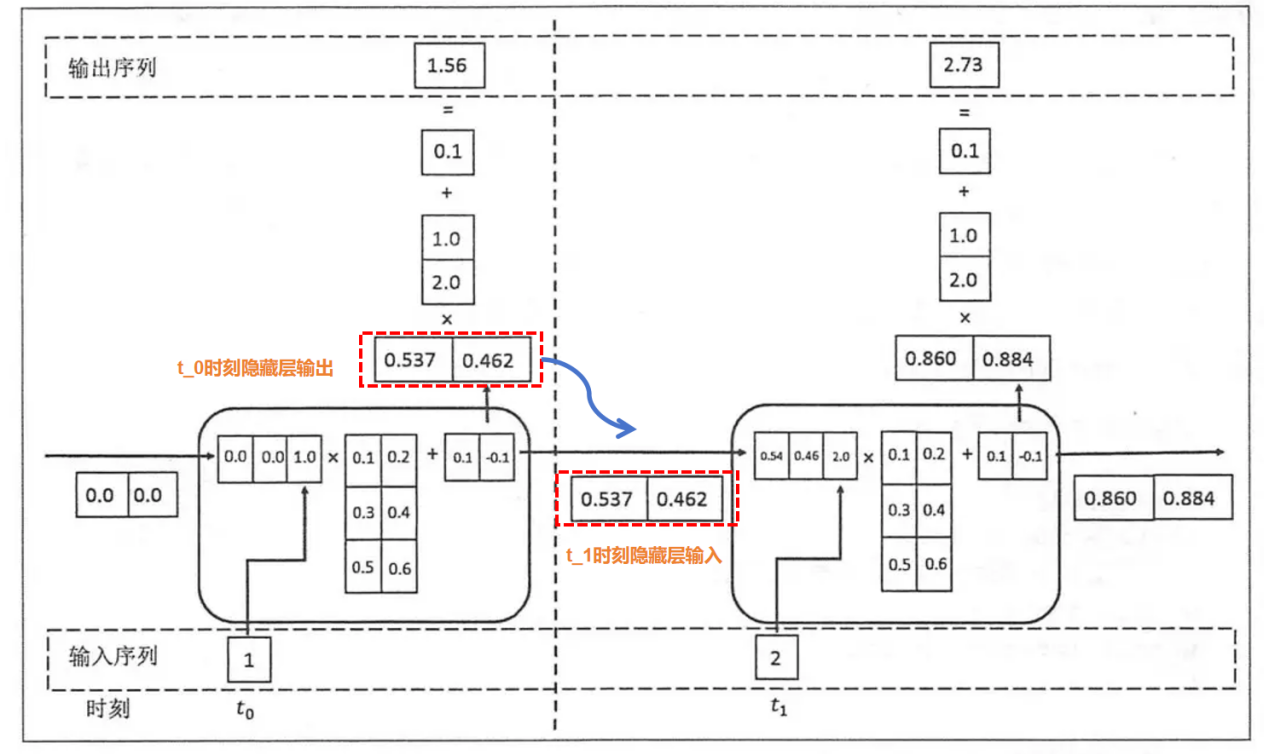
还有一个重要的一点，RNN在每个时间步都有输出，但在不同的任务中每一步的输出结果可能是我们不必关心的。例如在情感分析任务重，我们可能只关心最终的输出，而不是每个单词之后的情绪。

另外这里展开为几层，主要取决于我们的输入语句的单词数量，假设为 I love you，则展开为3层，如果是l love you very much，则展开为5层，每一个输入对应一个单词。

上图中RRN循环层被展开为3层，我们将其转化为从左到右的长神经网络，这类似于之前我们学习到的前馈神经网络，只是前馈神经网络的数据流向均为一个方向。

**“每一时刻的输出都跟当前时刻的输入和上一时刻的输出有关”**

这句话怎么理解呢？大家可以从下面这张图中进行了解，这里也感谢一下这张图的原作者。



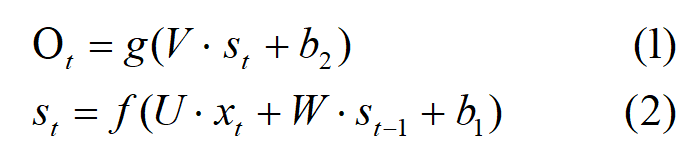
细心的同学可能已经发现，在RNN展开的过程中，U,V和W一直没有变化，这意味着在模型训练的时候这些参数都是共享的，没有发生变化。

至于此，我们就了解了RNN的结构了。

1. **进一步理解RNN**

基于展开的RNN结构我们可以得到下述公式：



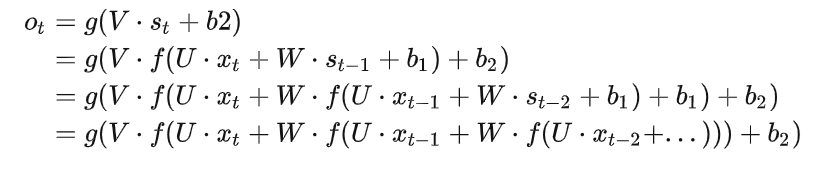


其中：

1. g为输出层非线性激活函数，比如常见的tanh和ReLU函数
2. f为隐层层非线性激活函数，比如常见的softmax激活函数
3. b1,b2均为相应的配置值。

从这个公式也能看得出，**每一时刻的输出都跟当前时刻的输入和上一时刻的输出有关。**

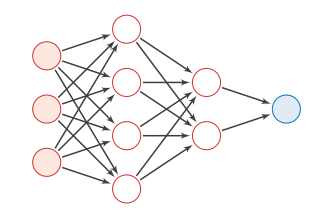
我们将（2）式带到（1）式中：



可以发现，当前时刻的输出包含了历史信息，这说明循环神经网络对历史信息进行了存储，但是这个历史信息不知道是有用的还是无用的，后面会讲述到LSTM和注意力机制我们再聊这个话题。**上述计算过程也即RNN的前馈计算过程。**

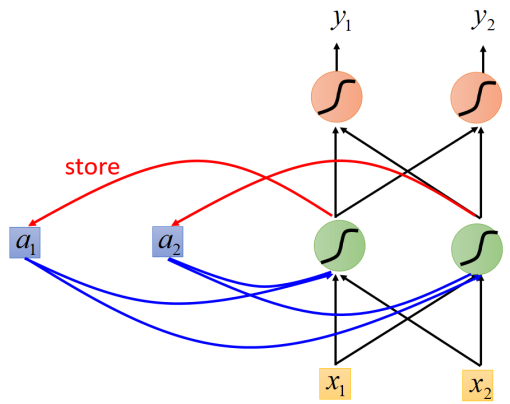
1. **RNN前馈计算案例**

如果我们回到：



这种网络结构下的RNN结构，RNN的输出该是什么样的呢？感谢李宏毅老师的深度学习教程。

假设现在我们的网络结构如下所示：



其中设置：



即权重参数为1，且不含配置项b，初始化a的值为[0,0]。

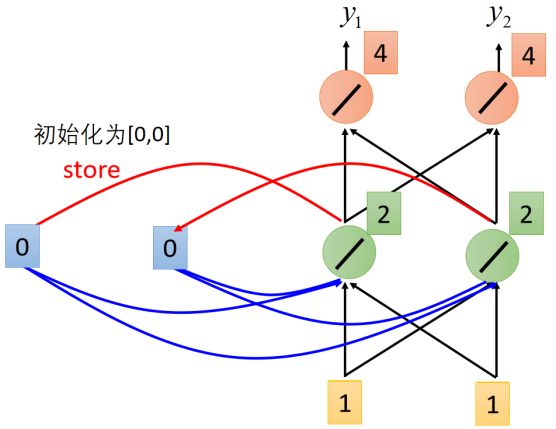
假设激活函数均为Liner线性激活函数，且3个假设输入为：



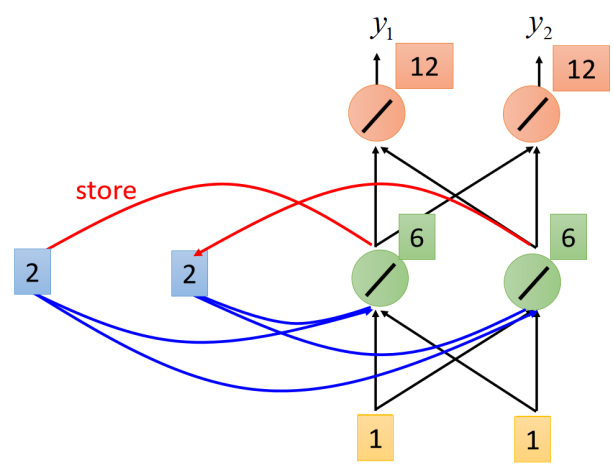
1. 则在输入的时候，有：

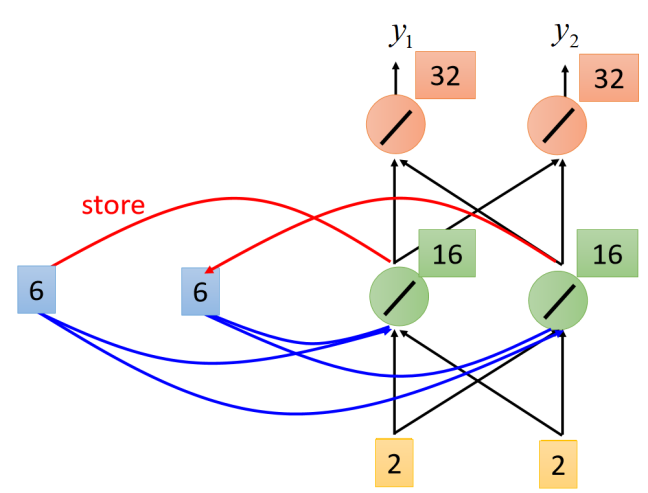
即在输入[1,1]的时候，得到隐藏层的a数值为[2,2]，输出结果Y为[4,4]



1. 则再次输入的时候，按照计算方式有：



1. 则输入的时候，按照计算方式有：



最终得到的输出结果为

如果使用代码计算可以如下所示：

import numpy as np  
  
data = [[1,2],[3,4]]  
x = [[1],[1]]  
  
A = np.mat(data)  
X = np.mat(x)  
  
print(isinstance(A,np.matrix))  
print(isinstance(X,np.matrix))  
  
print(A.shape)  
print(X.shape)  
  
print(np.dot(A,X))  
print(np.dot(A,X).shape)

通过上述流程，我们将RNN的计算流程就进行了完整的体验，显然每一时刻的输出结果都与上一时刻的输入有着非常大的关系，在实际的过程中，如果我们将输入的数据进行打乱，那么实际的输出也就会存在很大的不同，**也就是说RNN是一种输入序列敏感的神经网络**。

1. **总结：**

这篇文章和大家一起学习了RNN的网络，RNN具有一定的记忆功能，RNN对具有序列特性的数据非常有效，RNN中输入层到隐藏层，隐藏层到输出层共享相同的参数，这大大减少了我们需要学习的参数总数。

1. **参考资料：**
2. https://www.jianshu.com/p/c0a2e3984128
3. <https://zybuluo.com/hanbingtao/note/541458>
4. 李宏毅-深度学习教程

Pytorch实现：https://zhuanlan.zhihu.com/p/259594151

https://www.jianshu.com/p/c0a2e3984128