[译]Recurrent Neural Networks Totorial part1

神经网络

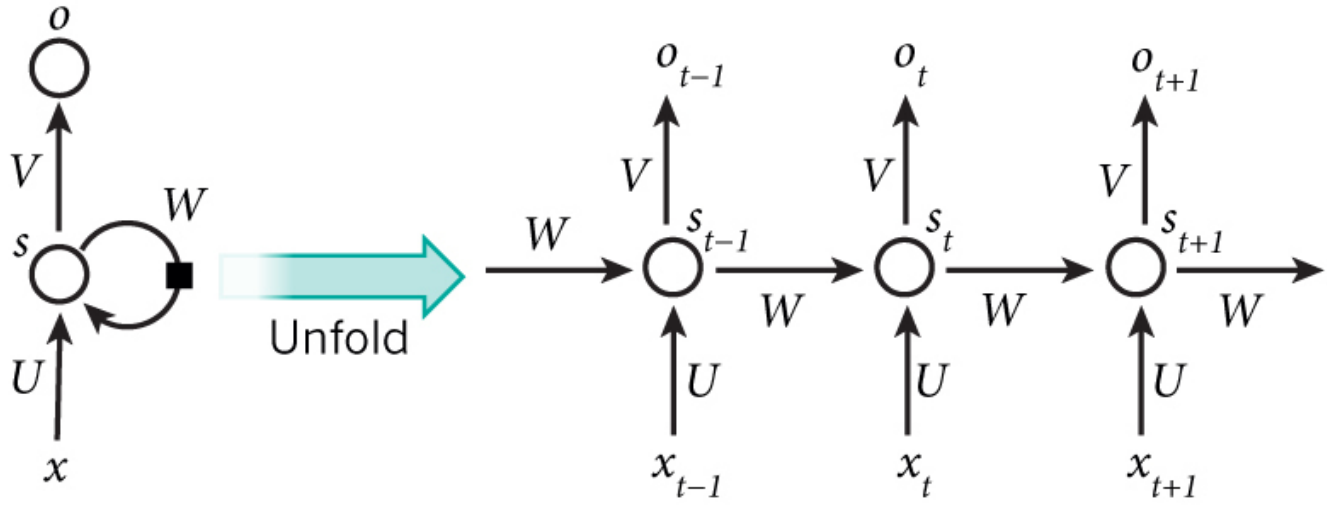
Recurrent Neural Networks(RNNs)作为一个很受欢迎的模型，已经在许多NLP任务展现了巨大的潜力。但是尽管他们最近很受欢迎，我还是仅找到一些很少的关于RNN如何工作，怎样实现他们的资料。这就是这个教程的目的。这是一个多系列的教程我计划包含如下部分：

1. Introduction to RNNs (this post)
2. Implementing a RNN using Python and Theano
3. Understanding the Backpropagation Through Time (BPTT) algorithm and the vanishing gradient problem
4. Implementing a GRU/LSTM RNN

在教程中我们会实现 recurrent neural network based language model。这个语言模型的应用有2个部分：第一，它允许我们基于现实世界的句子(sentences)给任意的句子(sentences)打分.这给我们提供一个度量语法和语义正确性的方法。这样的一个典型模型被用做机器翻译系统上。第二，一个可以生成新文本的语言模型（我认为这是非常酷的一个应用）。在莎士比亚文集(Shakespeare)上训练语言模型，生成莎士比亚文体的文本。Andrej Karpathy描述了一个基于RNNs的字符级的语言模型。
我假设你对基本的神经网络有些熟悉。如果不，那你可能回过头去看看Implementing A Neural Network From Scratch，这会知道你理解和实现non-recurrent networks。

什么是RNNs？

RNNs就是利用序列信息。在传统的神经网络里，我们假设所有的输入（和输出）都是相互独立的。但是对很多任务来说这是一个很坏的主意。如果你想预测一个句子中的下一个词，你最好知道之前的词。RNNs被叫做recurrent因为它们在序列中的每个元素都完成相同的任务，输入都依赖于前一步的计算。对RNNs的其他理解是把它们看作是一个能获取之前计算信息的记忆（"memory"）。理论上的RNNs能利用任意长度的序列信息，但是实际中它们受限于很少的几步。下面是一个典型的RNN：



A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

上面的图展示了RNN展开为一个全网络。通过展开我们简单的写出了完整序列的网络。例如，如果这个序列只有5个词，网络就展开成5层神经网络，每一次一个单词。RNN中的公式计算如下：

- x_t是在时间步t的输入，例如，x_1可以是one-hot表示的序列中的第二个词
- s_t是隐藏状态在时间步t。这是网络的"memory"，s_t基于前面隐藏状态和当前输入步计算:$s_t = f(Ux_t + Ws_{t-1})$.函数f通常采用非线性的$tanh$和$ReLU$.s_{-1}是需要计算的第一个隐藏状态，最典型的初始化是全为0。
- o_t是时间步t的输出。例如，如果你希望预测句子中的下一个词，它可能是一个向量的概率:$o_t = softmax(Vs_t)$

这里有一些需要注意的事：

- 你可以把隐藏状态s_t认为是一个网络的记忆。s_t获取之前所有时间步的信息。输入o_t单独

基于时间t的记忆.下面简单提及，在实际中会更加复杂一点，因为s_t不能从之前时间步获取太多信息.

- 不同于传统的深度神经网络，在每一层使用不同的参数，RNN在所有步共享相同的参数（U, V, W）这反应了一个事实，我们在每一步都执行相同的任务，只是使用了不同的输入。这极大的减少的我们需要学习的参数数量。
- 上图的每一步的输出，根据任务可能是不需要的。例如，句子情感预测的时候，我们只关注最后的输出，而不是每个词的情感。相似的，我们可能不需要每次的输入。RNN的主要特点是它的隐藏状态，它获取序列的一些信息。

RNNs能做什么？

RNNs在许多NLP任务展现了巨大的成功。这个点来说我会提及最通用的RNNs模型LSTMs，更好的捕获长距离依赖。但是不用担心，LSTMs本质上和RNN做的是同样的事情，他们只是在计算隐藏状态上的方法不同。在后面我们会有更详细的LSTMs介绍。这里列举一些RNNs在NLP的一些应用。

语言模型和生成模型

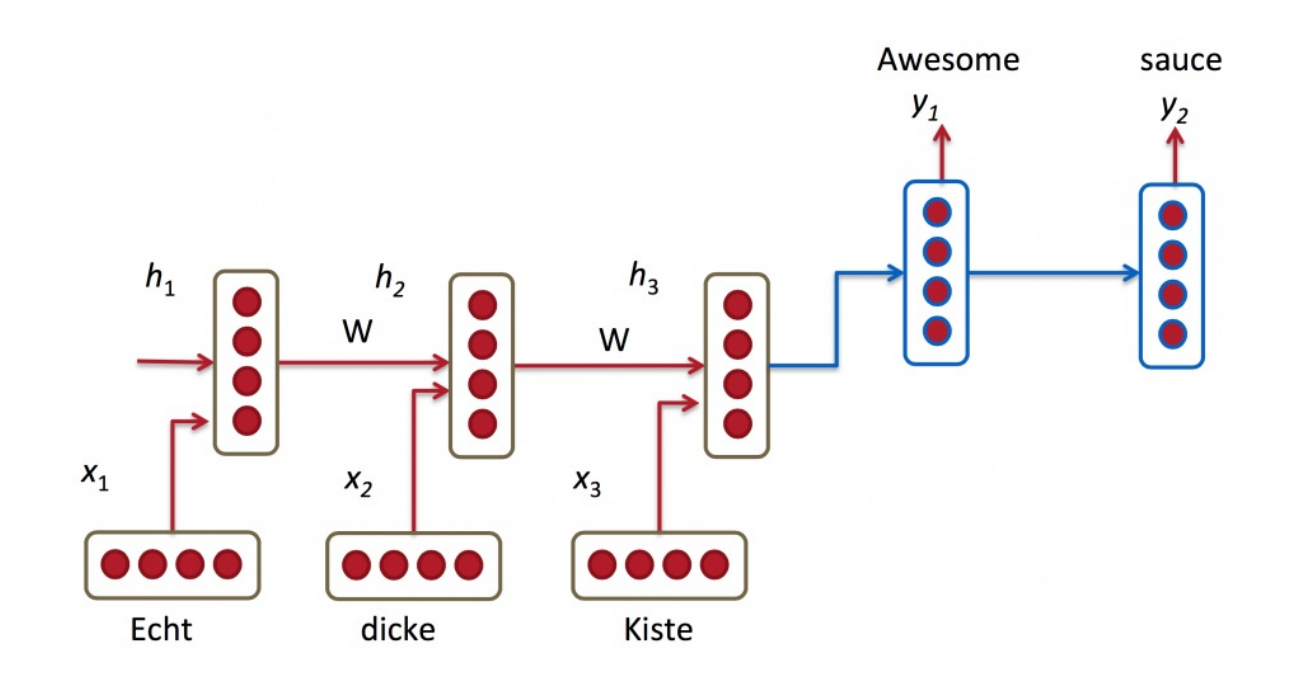
给定一个序列，我们希望给定前面的词预测每一个词的概率。语言模型度量句子的相似，这对于机器翻译来说是很重要的（概率高的句子代表正确）。通过输出概率中抽样生成新文本，预测下一个单词我们可以得到一个生成模型。训练数据决定了我们生成的东西。在语言模型里面我们的输入是典型的单词序列(例如使用one-hot向量编码)，我们的输入是序列的预测词。当训练网络时我们设置$o_t = x_{t+1}$是我们想要的时间步t的输出也就是下一个实际的词。
关于语言模型和文本生成的研究论文：

- Recurrent neural network based language model
- Extensions of Recurrent neural network based language model
- Generating Text with Recurrent Neural Networks

机器翻译

机器翻译类似于语言模型，从原始语言（eg.德语）输入的词序列，我们希望输入目标语言

(eg. 英语)的词序列.一个关键的不同在于我们的输出只有在完全输入之后才能看见，因为我们翻译的句子的第一词可能需要捕获完整的输入序列.



RNN for Machine Translation. Image Source:http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

关于机器翻译的研究论文：

- A Recursive Recurrent Neural Network for Statistical Machine
- Translation Sequence to Sequence Learning with Neural Networks
- Joint Language and Translation Modeling with Recurrent Neural Networks

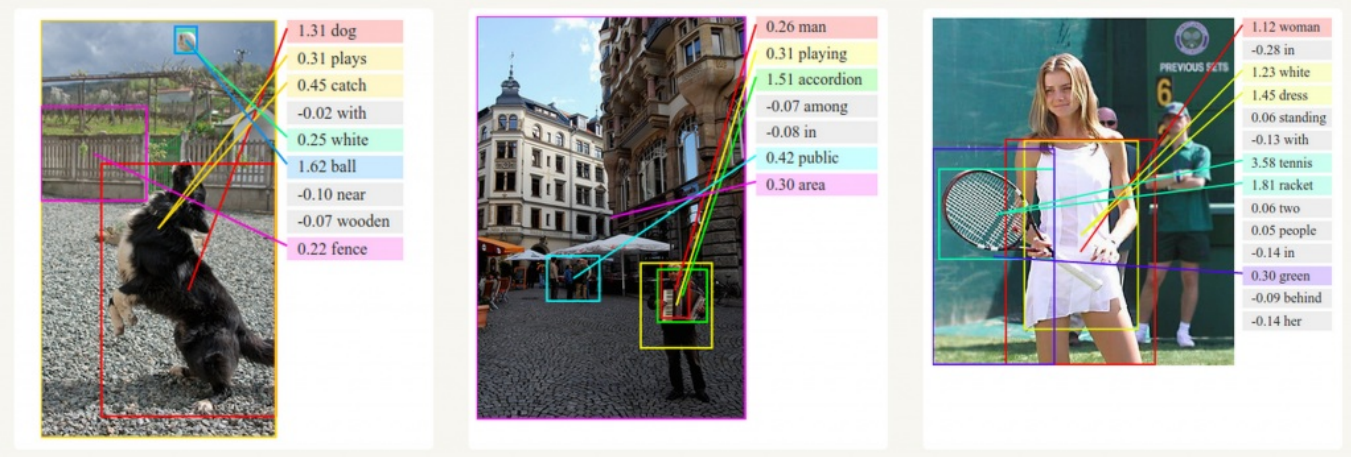
语音识别

给定声波的语音信号序列，我们可以一起预测语音片段的序列的概率
语音识别的研究论文：

- Towards End-to-End Speech Recognition with Recurrent Neural Networks

生成图像描述

结合卷积神经网络，RNNs被用作无标签图像的描述生成。很神奇它是如何工作的。这个联合模型使用图像中的特征生成词。



Deep Visual-Semantic Alignments for Generating Image Descriptions. Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

训练RNNs

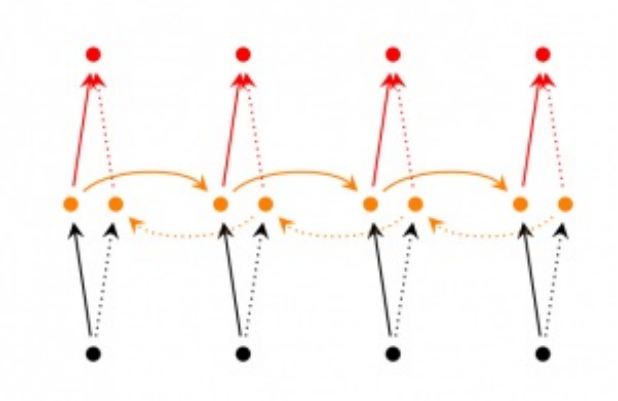
训练RNN和训练传统的神经网络很相似。我们同样使用后向传播算法，但是有一些曲折。因为参数在所有时间步共享，梯度在每一次输触不仅依赖于计算当前时间步，也需要前面的时间步。例如，为了计算时间步$t = 4$的梯度，我们需要时间步3和梯度求和。这被称作定时后向传播(Backpropagation Through Time).如果这对你还没有一个整体的概念，不要担心，后面我们会有更详细的介绍。现在，只需要知道原始的RNNs使用BPTT训练有长距离依赖的问题。这主要由于梯度爆炸和梯度消失问题引发。存在一些工具来处理这些问题，某些类型的RNNs（LSTMs）是专门设计来规避这个问题的。

RNN 扩展

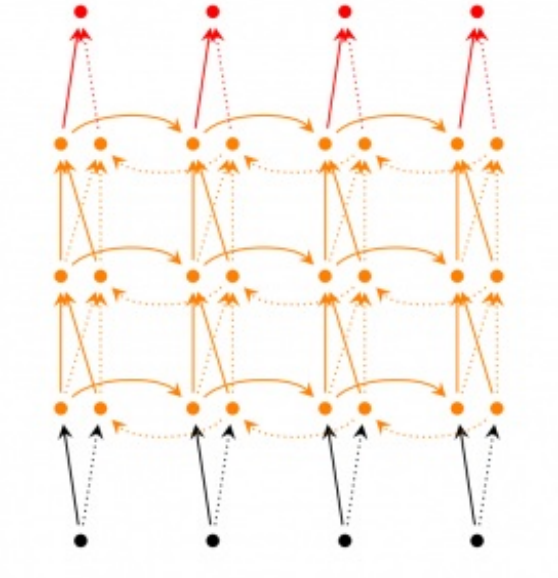
多年以来研究者们设计更加复杂的RNNs来处理原始RNN模型的缺陷。我们会在后面的覆盖更多的内容，但是在这个部分我想给你一个简介使你能熟悉模型的分类。

Bidirectional RNNs（双向RNN）：思想基于在时间步t的输出不仅依赖于序列中前面的元素也依赖于未来的元素。例如，预测一个序列中的缺失值，你需要考虑左右上下文。

Bidirectional RNNs是很简单的。他们只是2个RNNs堆叠而成。输出基于两个RNNs的隐藏层。



Deep (Bidirectional) RNNs （深度RNNs）:类似于双向RNNs，只是现在每一个时间步有多个层。实践中这给我更好的学习能力(但是也需要大量的训练数据)



LSTM network：最近很流行不同于我们上面谈论过的模型。LSTMs并不是基础架构上不同于RNNs，而是在计算隐藏状态的不同。LSTMs中的记忆叫做cell，可以把它们看作是一个使用前一个状态h_{t-1}和当前输入x_t的一个黑盒。在内部，cells决定该保留（擦除）那些记忆。然后联合前面状态，当前记忆和输入。事实证明这种类型的神经元能有效处理长距离问题。LSTMs一开始很让人困惑，但是如果你有兴趣了解更多这是一个优秀的解释

结论

到目前为止。我希望你基本明白了关于什么是RNNs以及他们能做什么。在下一个帖子里，我们会用Python和Theano实现RNN的第一个版本。

> 原文地址