2.0 概览

2.0.1 课程目标

本课程的目标是理解基于时间的反向传播算法（Backpropagation Through Time algorithm）来训练LSTMs。完成本课程之后，你将会知道：

- 基于时间的反向传播算法是什么？以及它涉及的多层感知机网络所使用的反向传播训练算法。
- 引起Truncated Backpropagation Through Time需要的动机，深度学习中训练LSTMs时最广泛的使用的变体。
- 一种用于考虑如何通过时间和时间来研究反向传播的规范，以及用于研究和深度学习库中的正则表达式。

2.0.2 课程概要

本课程分为6个部分，它们是：

1. 反向训练算法；
2. 展开循环神经网络；
3. 基于时间的反向传播；
4. 截断基于时间的反向传播；
5. 截断BPTT的配置；

6. TBPTT的Keras实现。

让我们开始吧！

2.1 反向传播训练

反向传播是指两件事情：

- 计算导数的数学方法和链式求导规则的使用；
- 更新网络权值以最小化误差的训练算法；

这是我们在本课中使用的后向传播算法的理解。反向传播算法训练的目的是修改神经网络的权重，以使网络输出的误差与响应于相应输入的一些预期输出相比最小化。这是一个监督学习算法，允许网络对所犯特定错误进行校正。一般算法如下：

1. 提出一种训练输入模式，并通过网络传播训练以获得输出；
2. 将预测输出与预期输出进行比较，并计算误差；
3. 计算误差相对于网络权重的导数；
4. 重复。

2.2 展开循环神经网络

循环神经网络的一个简单概念是一种神经网络，它从先前的时间步长中获得输入。我们可以用图来阐明这一点：

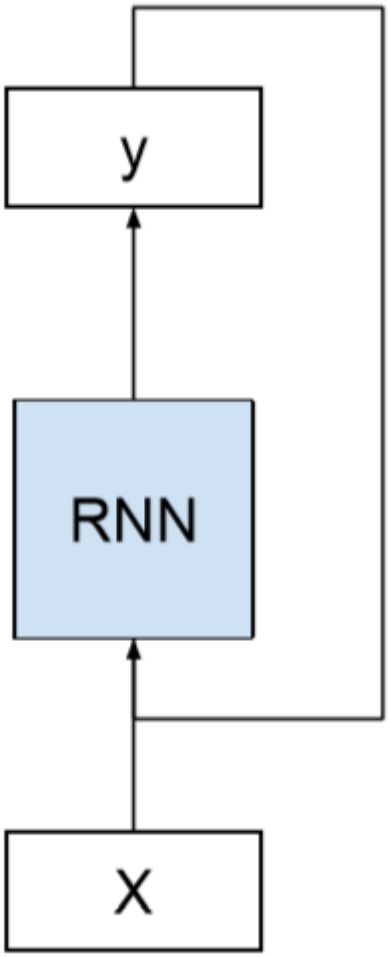

图2.1 循环神经网络的一个简单示例

RNNs用于拟合，并在多时间步长上进行预测。随着时间步长的增加，具有递归连接的简单图开始失去所有意义。我们可以通过在输入序列上展开或者展开RNN图来简化模型。

> 可视化RNNs的一个有用的方法是考虑通过沿着输入序列“展开”网络形成的更新图。
>
> ——《用循环神经网络实现监督序列标签》，2008

2.2.1 展开向前传递

考虑下列的情况，我们有多个时间步长的输入`x(t),x(t+1),...`，多个时间步长的内部状态`u(t),u(t+1),...`，以及多个时间步长的输出`y(t),y(t+1),...`。我们可以将网络示意图展开成为没

有任何环的图，如下所示：

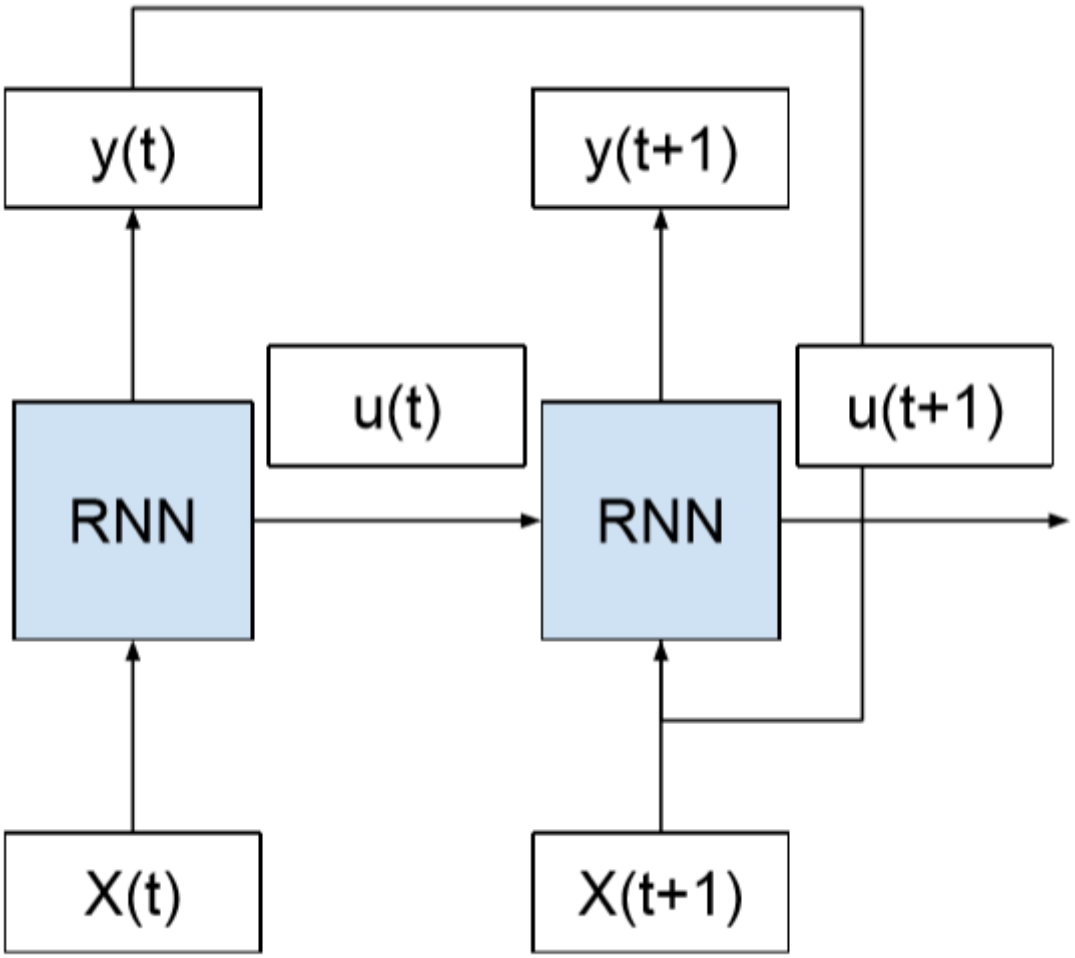

图2.2 展开循环神经网络的例子

我们可以看到，我们除去了环，并且从先前的时间步长输出`(y(t))`和内部状态`(u(t))`作为输入，用于处理下一个时间步长。这个概念中的关键是网络（RNN）在展开的时间步长之间不改变。特别地，对于每个时间步长使用相同的权值，它只是不同的输出和内部状态。这样，就好像在输入序列中的每个时间步长复制了整个网络（拓扑和权重）。

我们可以把这个概念进一步地深化，在那里每个网络副本可以被看做是同一前馈神经网络的附加层。较深的层作为输入的前一层的输出以及一个新的输入时间步长。这些层实际上是同一组权重的所有副本，并且从层到层更新内部状态，这可能是这种经常使用的类比的延伸。

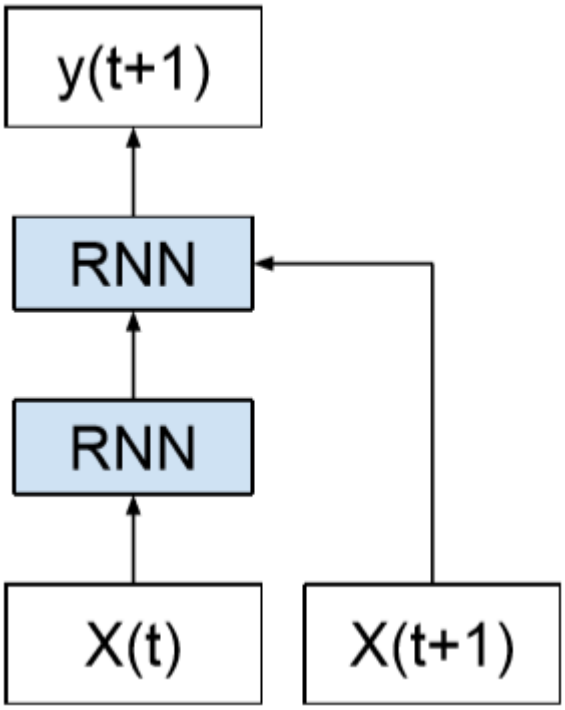

图2.3 以层为单位展开递归神经网络的例子

> RNNs一旦在时间上展开，时间可以被看做是非常深的前馈网络，其中所有的层共享相同的权重。
>
> ——Deep learning, Nature,2015

这是一个有用的工具，可视化有助于理解在前向传递中网络中发生了什么。它可能是也可能不是，网络是由深度学习库所的实现方式。

2.2.2 前向传播展开

网络展开的思想在循环神经网络实现向后传递方面发挥了更大的作用。

> 作为“通过时间的反向传播”的标准，网络随着时间的推移而展开，因此到达层的连接被视为来自先前的时间步长。
>
> — Framewise phoneme classification with bidirectional LSTM and other neural network architectures, 2005

重要的是，对于给定的时间步长，误差的反向传播取决于在先前时间步长的网络的激活。以这种方式，向后传递需要展开网络概念化。误差被传播回序列的第一输出时间步长，从而可以计算误差梯度，并且可以更新网络权重。

> 像标准反向传播，“基于时间的反向传播”包括重复应用链式规则。微妙的是，对于循环神经网络，损失函数依赖于隐藏层的激活，不仅依赖于其在输出层上的隐藏激活，还取决于隐层在下一个时间步长上的隐层的激活。
>
> — Supervised Sequence Labelling with Recurrent Neural Networks, 2008.

展开循环神经网络图也引入了额外的关注点。每个时间步长都需要一个新的网络副本，而网络又需要更多的内存，特别是对于具有数千或者数百万个权值的大型网络。随着时间步长攀升到数百，训练大型循环网络的内存需求会迅速膨胀。

> ...它需要根据输入序列的长度展开RNNs。通过将RNN N次展开，网络内的神经元的每一次激活被复制N次，这会消耗大量的存储器，特别是当序列很长的时候。这阻碍了小规模的在线学习或者适应的实施。此外，这种“完全展开”使得多个序列并行训练，如图形处理单元（GPU）等共享内存模型。
>
> ——Online Sequence Training of Recurrent Neural Networks with Connectionist Temporal Classification, 2015.

2.3 基于时间的反向传播算法

基于时间的反向传播算法（或叫做BPTT），是反向传播训练算法在递归神经网络中的应用。在最简单的情况下，循环神经网络被显示出了每一个时间步长的一个输入，并预测一个输出。从概念上讲，BTPP通过展开所有时间步长来工作。每个时间步长具有一个输入时间步长、一个网络拷贝和一个输出。然后对每个时间步长计算和累计误差。网络被回滚，权重被更新。我们可以将算法总结如下：

1. 向网络呈现输入和输出对时间步长的序列；
2. 展开网络，然后计算并累计每个时间步长的误差
3. 卷起网络并更新权重；
4. 重复。

BPTT随着时间步长的增加而计算变得昂贵。如果输入序列由数千个时间步长组成，那么这将是单个权重更新所需的导数的数目。这会导致梯度消失或者梯度爆炸（变成0或者溢出），并使学习缓慢或者模型技能嘈杂。

> BPTT的主要问题之一就是但参数更新的成本太高，这使得不可能使用大量的迭代。
>
> — Training Recurrent Neural Networks, 2013

解决梯度消失和梯度爆炸的一种方法是在执行权重更新之前限制时间步长的数量。

2.4 截断基于时间的反向传播算法

截断基于时间的反向传播算法，或者简称为TBPTT，是递归神经网络的BPTT训练算法的修改版本，其中序列被一次处理一个时间步长，并且周期性地为一个固定数量的时间步长执行更新。

> 截断BPTT...在一个时间的一个时间步长处理时间序列，并且每 `k1` 时间步长，在 `k2` 时间步长它运行BPTT，因此，如果 `k2` 很小，参数更新可以是很便宜的。因此，它的隐藏状态已经被暴露于很多的时间步长，因此可能包含关于很久之前的有用信息，这将会被选择性的利用。
>
> — Training Recurrent Neural Networks, 2013

我们可以总结算法如下：

1. 提出了一系列的`k1`时间步长的网络输入和输出对；
2. 打开网络，然后计算并累计`k2`时间步长的误差；
3. 卷起网络并更新权重；
4. 重复。

TBPTT算法需要考虑两个参数： `k1`：更新之间的正向传递的时间步数。一般来说，这个是一个缓慢或者快速的训练，因为经常进行权重更新。 `k2`：应用BPTT的时间步数。一般来说，它应该足够大，以捕获网络中的时间结构以学习网络。过大的值导致梯度消失。 截断BPTT的一个简单应用是实现将`k1`设置为序列长度，并调整`k2`以训练速度和模型技能。

2.5 截断BPTT的配置

我们可以把事情更加向前推进一步，用一种符号来帮助更好地理解BPTT。在它们处理的BPTT的时候命名为“An Ecient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories”，Williams和Peng设计了一种符号来铺货截断和未截断的配置，例如：`BPTT(h)`和`BPTT(h;1)`。

我们可以沿用这个符号，并使用Sutskever的`k1`和`k2`参数（如上）。使用这种符号，我们可以得出一些标准或者常用的方法。注：这里`n`指的是输入序列中的时间步长总数。

- `BPTT(n,n)`：更新实在序列结束时在序列中的所有时间步长进行的（例如：经典的BPTT）.
- `BPTT(1,n)`：时间步长一次处理一次，接着更新一个更新，涵盖迄今为止所见的所有时间步长（例如，Williams和Peng的经典TBPTT）。
- `BPTT(k1,1)`：网络可能没有足够的时间背景来学习，很大程度上依赖于内部状态和输入。
- `BPTT(k1,k2)`：其中，`k1<k2<N`：每个序列执行多个更新，可以加速训练。
- `BPTT(k1,k2)`：其中`k1=k2`：一个共同的配置，其中一个固定的时间步长用于前向和后向时间步长（例如10s到100s）。

我们可以看到，所有配置是`TBPTT(n,n)`上的一种变化，本质上试图以更快的训练和更稳定的结果来近似相同的效果。文献中报道的规范TBPTT可以被认为是`TBPTT（k1，k2）`，`k1=k2=k`和`k<=N`，并且所选择的参数小（几十到几百个时间步长）。这里，`k`是必须指定的单个参数。人们经常声称输入时间步长的序列长度应该限制在200～400之间。

2.6 Keras实现TBPTT

Keras深度学习lib库提供递归神经网络的训练TBPTT的实现。实现比上面列出的一般版本更受限制。特别地，`k1`和`k2`的值是相等的，并且是固定的。

- `TBPTT(K1,K2)`，其中`k1＝k2＝k`。

这是通过训练像LSTM这样的递归神经网络所需的三维输入来实现的。LSTM期望输入数据具有维度：样本、时间步长和特征。这是输入格式的第二个维度，即时间步长，它减少了用于序列预测问题的向前和向后传递的时间步长的数量。

因此，在为Keras编写序列预测问题的输入数据时，必须谨慎地选择时间步数。时间步长的选择将在两个方面：

- 在前向传播的过程中积累的内部状态。
- 用于更新后向传播上权重的梯度估计。

注意，默认情况下，网络的内部状态在每个批次之后被重置，但是通过使用所谓的状态LSTM并手动调用重置操作，可以实现对内部状态重置时的更明确的控制。以后再谈这个问题。

该算法的Keras实现本质上是未截断的，要求在训练模型之前直接对输入序列执行任何截断。我们可以认为这是手动截断的BPTT。Sutskever称这是一个朴素的方法。

> ...一种朴素的方法，将1000个长序列分成50个序列（例如）每个长度20，并将长度20的每个序列作为单独的训练案例处理。这是一种明智的方法，在实践中可以很好地工作，但是它对跨越20个以上的时间步长的时间依赖性是盲目的。
>
> — Training Recurrent Neural Networks, 2013

这意味着作为你的问题框架的一部分，你必须把长的序列分成足够长的子序列来捕获相关的上下文来进行预测，但是足够短，以便很好地训练网络。

2.7 扩展阅读

下面的章节提供了进一步阅读的一些资源。

2.7.1 书籍

- Neural Smithing, 1999.
- Deep Learning, 2016.
- Supervised Sequence Labelling with Recurrent Neural Networks, 2008.

2.7.2 研究论文

- Online Sequence Training of Recurrent Neural Networks with Connectionist Temporal Classification, 2015.
- Framewise phoneme classification with bidirectional LSTM and other neural network architectures, 2005.
- Deep learning, Nature, 2015.
- Training Recurrent Neural Networks, 2013.
- Learning Representations By Backpropagating Errors, 1986.
- Backpropagation Through Time: What It Does And How To Do It, 1990.
- An Ecient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories, 1990.

- Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity, 1995.

2.8 扩展

你想更深入地了解BPTT吗？本章节列出了本课程中的一些具有挑战性的扩展。

- 为新的学习者写一段关于BPTT的算法总结；
- 研究和描述使用上述符号在最近或显著的LSTM研究论文中使用的BPTT参数；
- 设计一个实验来调整BPTT的参与以用于序列预测问题；
- 研究和实现电子表格或者Python中单个存储单元的BPTT算法。

在网上发布你的扩展并与我分享链接。我很想知道你是怎么样想的！

2.9 总结

本课中，您发现了通过时间算法来训练LSTM序列预测问题的反向传播算法。特别地，您学到了：

- 什么是基于时间的反向传播算法以及它和感知机网络中反向传播训练算法是有着怎么样的联系；
- 导致需要截断基于时间的反向传播算法的动机，在深度学习中训练LSTMs模型最广泛使用的变体；
- 关于怎么样配置TBPTT的创新性思考，以及研究和机器学习库中用到的典型的配置。

接下来，您将发现如何准备用于LSTM的序列数据。