Lecture 07 Vanishing Gradients and Fancy RNNs

Lecture Plan

上节课我们学了**递归神经网络(RNNs)** 以及为什么它们对于语言建模(LM)很有用。今天我们将学习

- RNNs的 **问题** 以及如何修复它们
- 更复杂的 **RNN变体**

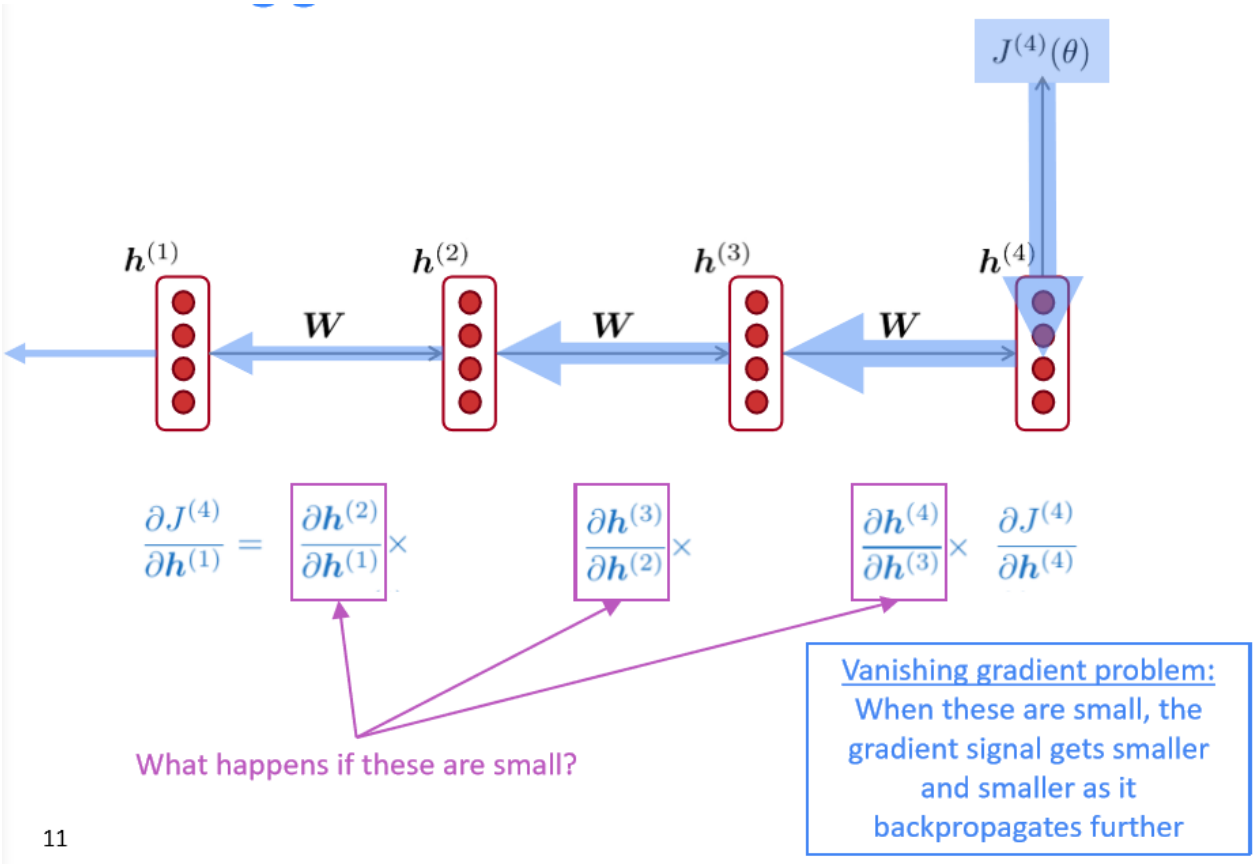
下一节课我们将学习

- 如何使用基于 RNN-based 的体系结构，即 **sequence-to-sequence with attention** 来实现 **神经机器翻译** (NMT)

Today's lecture

- 梯度消失问题 → 两种新类型RNN：LSTM和GRU
- 其他梯度消失（爆炸）的解决方案
  - Gradient clipping
  - Skip connections
- 更多花哨的RNN变体
  - 双向RNN
  - 多层RNN

Vanishing gradient intuition



- 当这些梯度很小的时候，反向传播的越深入，梯度信号就会变得越来越小

Vanishing gradient proof sketch

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right) \tag{1}$$

- 因此通过链式法则得到：

$$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} = \operatorname{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right) \boldsymbol{W}_h \tag{2}$$

- 考虑第 i 步上的损失梯度 $J^{(i)}(\theta)$，相对于第 j 步上的隐藏状态 $\boldsymbol{h}^{(j)}$

$$\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \prod_{j<t\leq i} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} \qquad \text{(chain rule)}$$

$$= \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \boldsymbol{W}_h^{(i-j)} \prod_{j<t\leq i} \operatorname{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right) \qquad \text{(value of } \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}\text{)}$$

If W_h is small, then this term gets vanishingly small as *i* and *j* get further apart

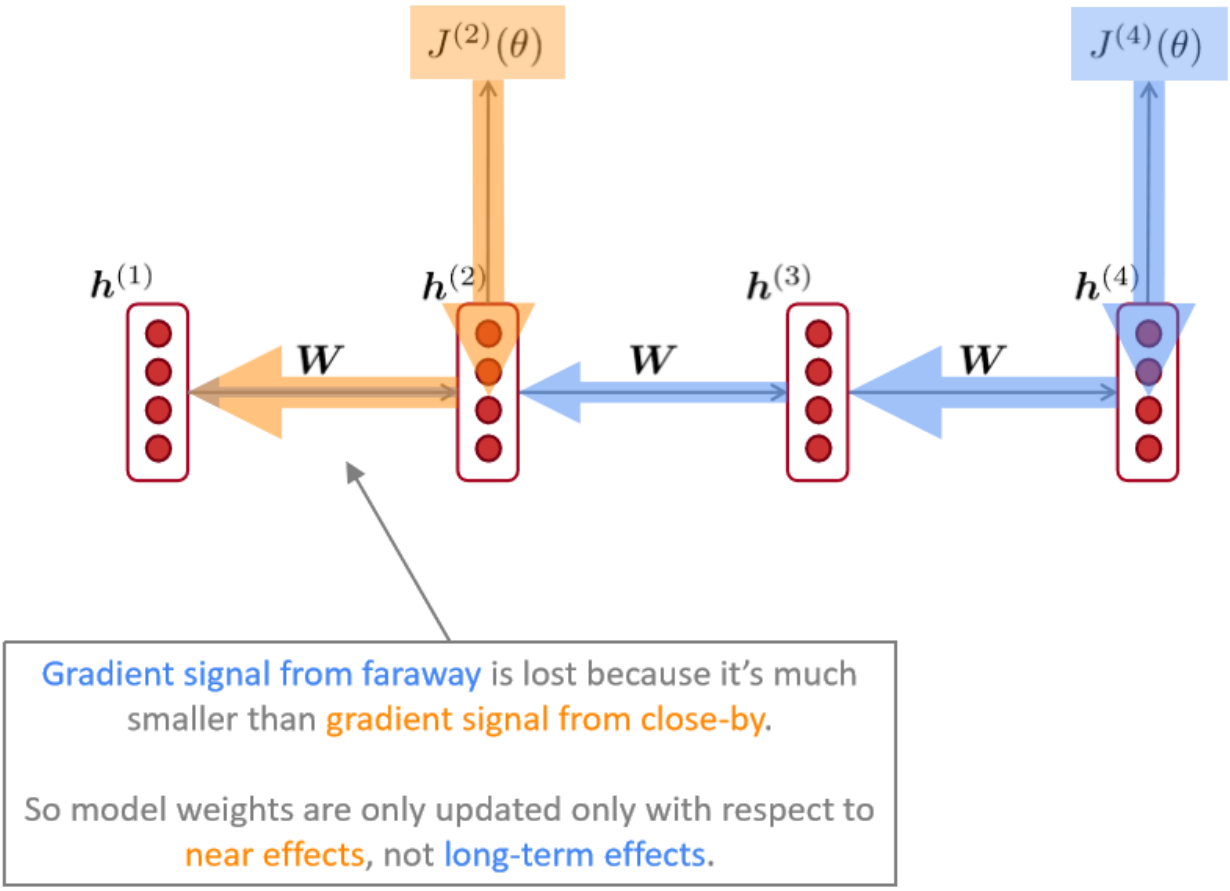
如果权重矩阵 W_h 很小，那么这一项也会随着 i 和 j 的距离越来越远而变得越来越小

- 考虑矩阵的 L2 范数

$$\left\|\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}}\right\| \leq \left\|\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}}\right\| \|\boldsymbol{W}_h\|^{(i-j)} \prod_{j<t\leq i} \left\|\operatorname{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right)\right\| \tag{3}$$

- Pascanu et al 表明，如果 W_h 的 **最大特征值 < 1**，梯度 $\|\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}}\|$ 将呈指数衰减
  - 这里的界限是1因为我们使用的非线性函数是 sigmoid
- 有一个类似的证明将一个 **最大的特征值 >1** 与 **梯度爆炸** 联系起来

Why is vanishing gradient a problem?



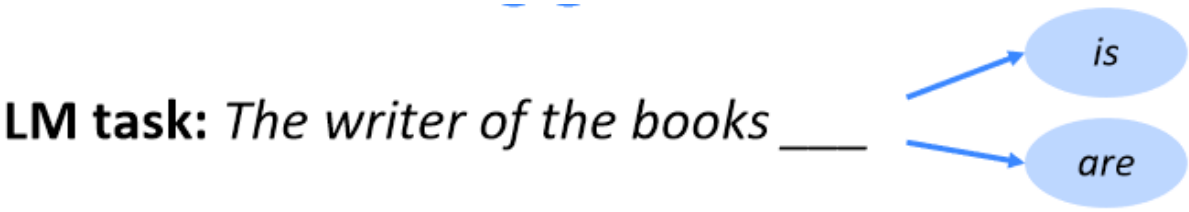
- 来自远处的梯度信号会丢失，因为它比来自近处的梯度信号小得多。
- 因此，模型权重只会根据近期效应而不是长期效应进行更新。
- 另一种解释： **梯度** 可以被看作是 **过去对未来的影响** 的衡量标准
- 如果梯度在较长一段距离内(从时间步 t 到 t+n)变得越来越小，那么我们就不能判断:
  - 在数据中，步骤 t 和 t+n 之间没有依赖关系
  - 我们用 **错误的参数** 来捕获 t 和 t+n 之间的真正依赖关系

Effect of vanishing gradient on RNN-LM

- 语言模型任务

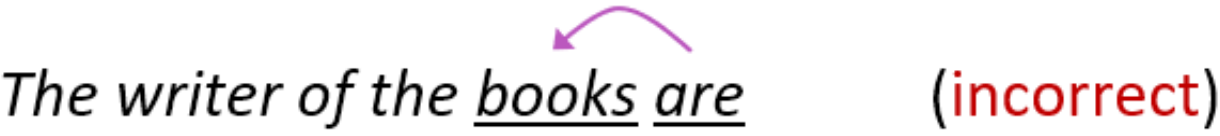
LM task: *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _____*

- 为了从这个训练示例中学习，RNN-LM需要对第7步的“tickets”和最后的目标单词“tickets”之间的依赖关系建模。
- 但是如果梯度很小，模型就 **不能学习这种依赖关系**
  - 因此模型无法在测试时 **预测类似的长距离依赖关系**



- **Correct answer** : The writer of the books is planning a sequel
- **语法近因**

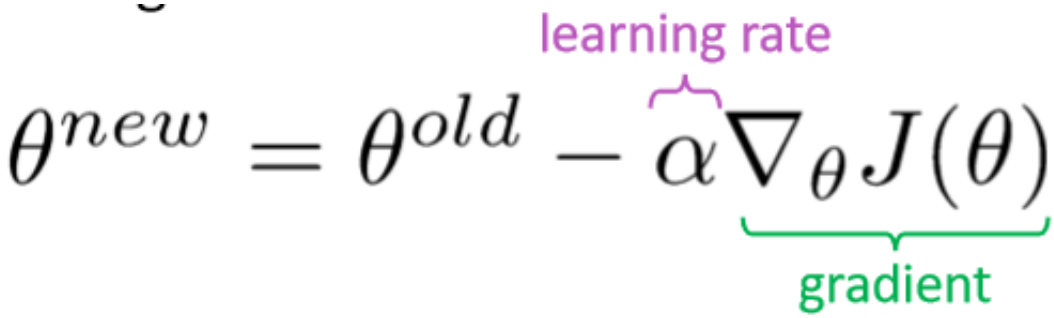
- **顺序近因**



- 由于梯度的消失，RNN-LMs更善于从 **顺序近因** 学习而不是 **语法近因** ，所以他们犯这种错误的频率比我们希望的要高[Linzen et al . 2016]

Why is exploding gradient a problem?

- 如果梯度过大，则SGD更新步骤过大



- 这可能导致 **错误的更新** ： 我们更新的太多，导致错误的参数配置(损失很大)
- 在最坏的情况下，这将导致网络中的 **Inf** 或 **NaN** (然后您必须从较早的检查点重新启动训练)

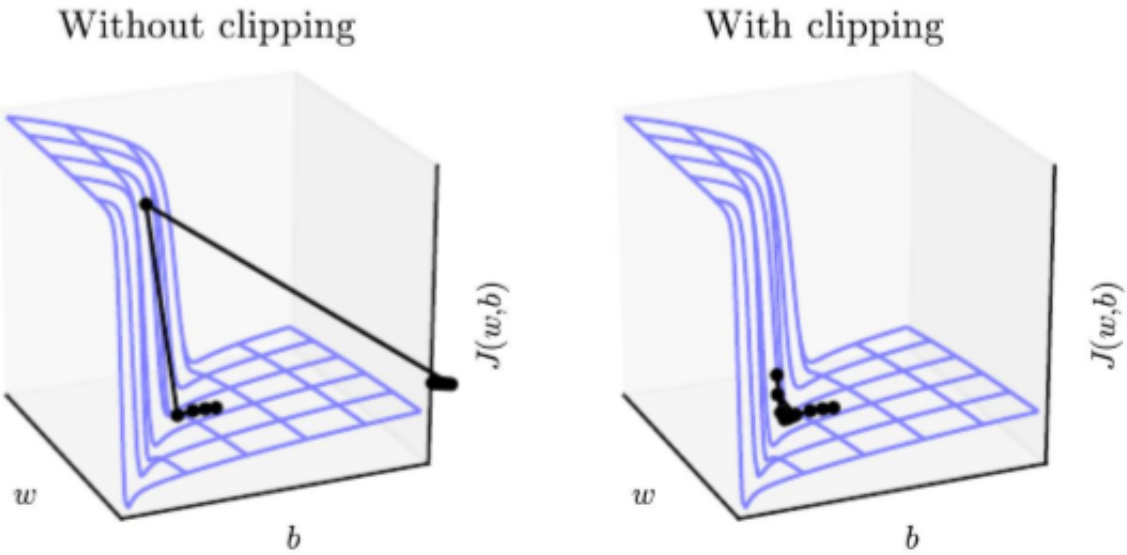
Gradient clipping: solution for exploding gradient

- **梯度裁剪** ： 如果梯度的范数大于某个阈值，在应用SGD更新之前将其缩小

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

- **直觉**： 朝着同样的方向迈出一步，但要小一点



- 这显示了一个简单RNN的损失面(隐藏层状态是一个标量不是一个向量)
- “悬崖”是危险的，因为它有陡坡
- 在左边，由于陡坡，梯度下降有两个非常大的步骤，导致攀登悬崖然后向右射击(都是胡浩的更新)
- 在右边，梯度剪裁减少了这些步骤的大小,所以效果不太激烈

How to fix vanishing gradient problem?

- 主要问题是RNN很难学习在多个时间步长的情况下保存信息
- 在普通的RNN中，隐藏状态不断被重写

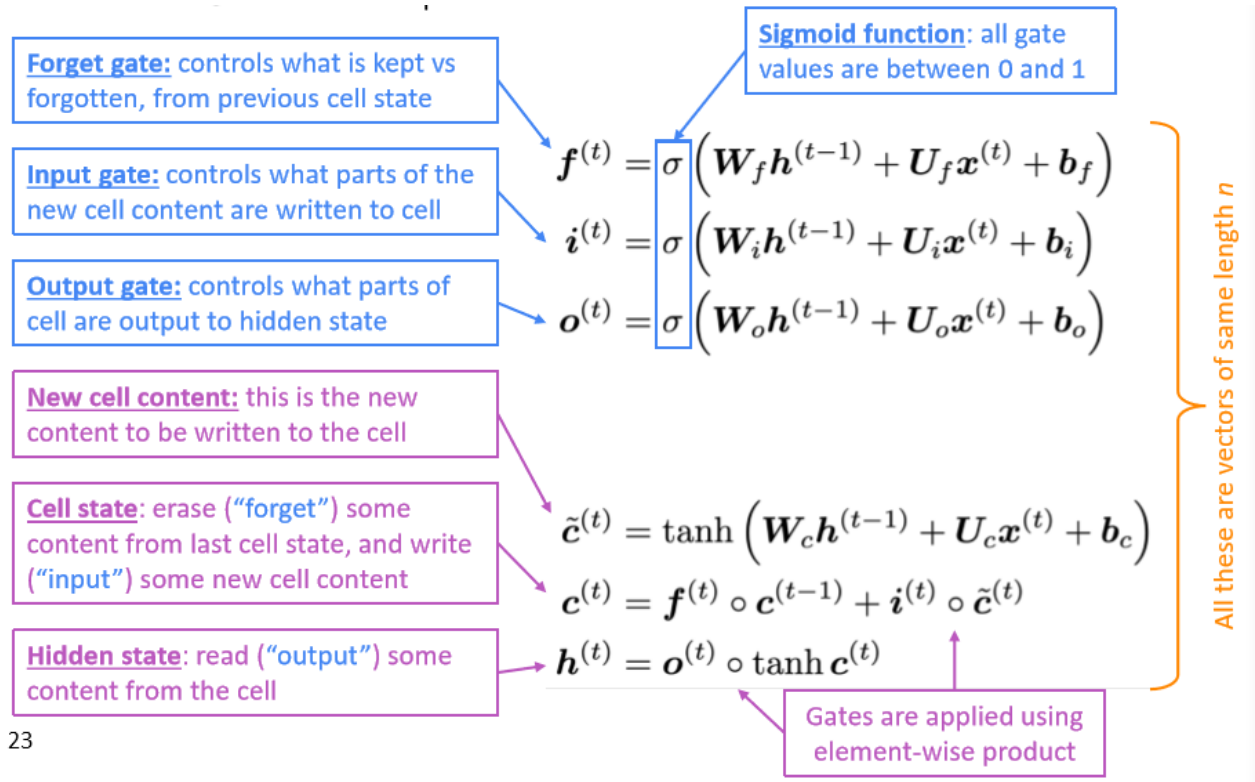
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}\right) \tag{4}$$

- 一个具有独立记忆的RNN怎么样?

Long Short-Term Memory (LSTM)

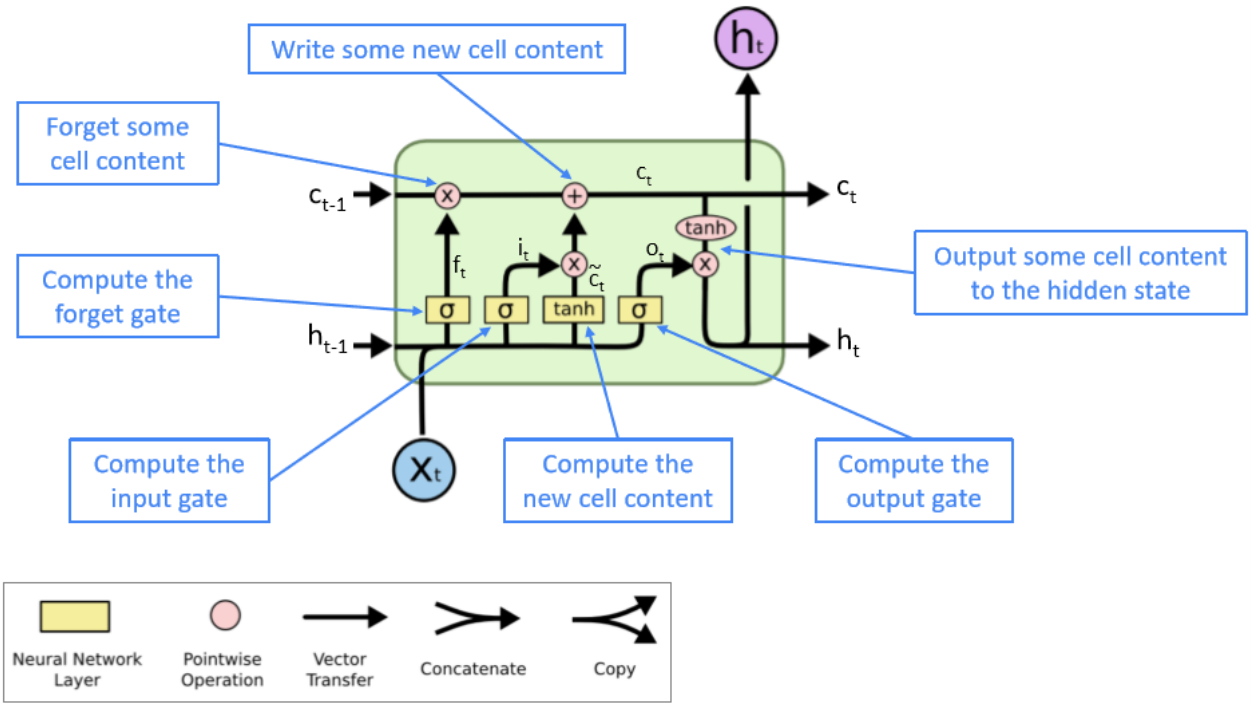
- Hochreiter和Schmidhuber在1997年提出了一种RNN，用于解决梯度消失问题。
- 在第 t 步，有一个隐藏状态 $h^{(t)}$ 和一个单元状态 $c^{(t)}$
  - 都是长度为 n 的向量
  - 单元存储长期信息
  - LSTM可以从单元格中删除、写入和读取信息
- 信息被 擦除 / 写入 / 读取 的选择由三个对应的门控制
  - 门也是长度为 n 的向量
  - 在每个时间步长上，门的每个元素可以打开(1)、关闭(0)或介于两者之间
  - 门是动态的：它们的值是基于当前上下文计算的

我们有一个输入序列 $x^{(t)}$ ，我们将计算一个隐藏状态 $h^{(t)}$ 和单元状态 $c^{(t)}$ 的序列。在时间步 t 时



- 遗忘门：控制上一个单元状态的保存与遗忘
- 输入门：控制写入单元格的新单元内容的哪些部分
- 输出门：控制单元的哪些内容输出到隐藏状态
- 新单元内容：这是要写入单元的新内容
- 单元状态：删除(“忘记”)上次单元状态中的一些内容，并写入(“输入”)一些新的单元内容
- 隐藏状态：从单元中读取(“output”)一些内容
- Sigmoid函数：所有的门的值都在0到1之间
- 通过逐元素的乘积来应用门
- 这些是长度相同的向量

你可以把LSTM方程想象成这样：



How does LSTM solve vanishing gradients?

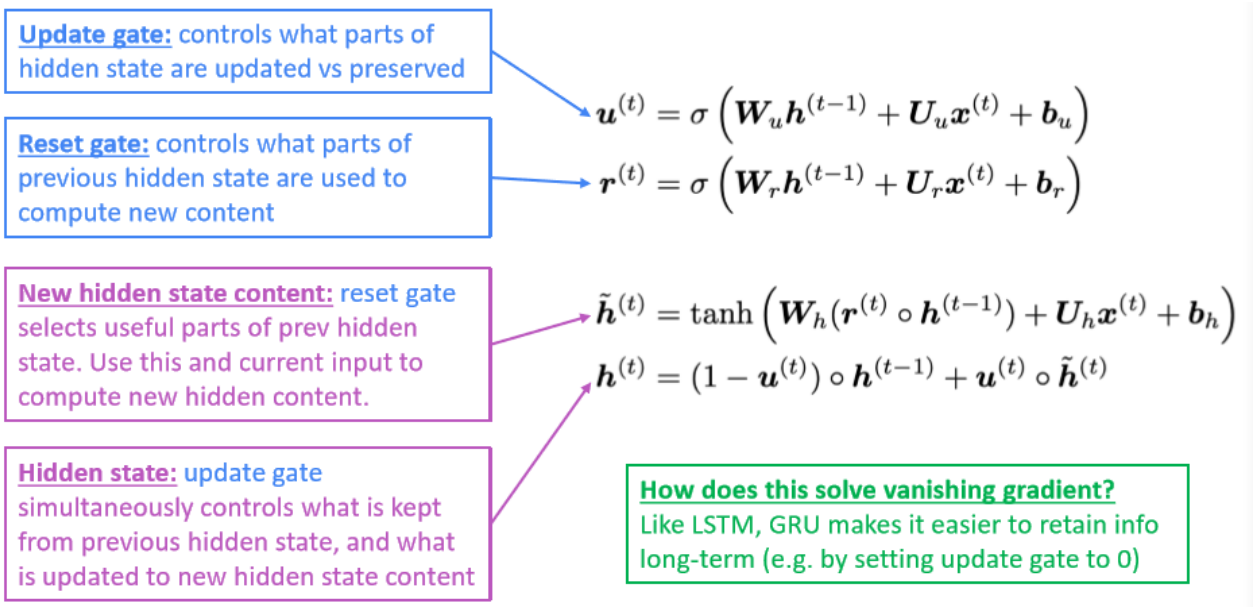
- RNN的LSTM架构更容易保存许多时间步上的信息
  - 如果忘记门设置为记得每一时间步上的所有信息，那么单元中的信息被无限地保存
  - 相比之下，普通RNN更难学习重复使用并且在隐藏状态中保存信息的矩阵 W_h
- LSTM并不保证没有消失/爆炸梯度，但它确实为模型提供了一种更容易的方法来学习远程依赖关系

LSTMs: real-world success

- 2013-2015年，LSTM开始实现最先进的结果
  - 成功的任务包括：手写识别、语音识别、机器翻译、解析、图像字幕
  - LSTM成为主导方法
- 现在(2019年)，其他方法(如Transformers)在某些任务上变得更加主导。
  - 例如在WMT (a MT conference + competition)中
  - 在2016年WMT中，总结报告包含“RNN”44次
  - 在2018年WMT中，总结报告包含“RNN”9次，“Transformers” 63次

Gated Recurrent Units (GRU)

- Cho等人在2014年提出了LSTM的一个更简单的替代方案
- 在每个时间步 t 上，我们都有输入 $x^{(t)}$ 和隐藏状态 $h^{(t)}$ (没有单元状态)



- 更新门：控制隐藏状态的哪些部分被更新，哪些部分被保留
- 重置门：控制之前隐藏状态的哪些部分被用于计算新内容
- 新的隐藏状态内容：重置门选择之前隐藏状态的有用部分。使用这一部分和当前输入来计算新的隐藏状态内容
- 隐藏状态：更新门同时控制从以前的隐藏状态保留的内容，以及更新到新的隐藏状态内容的内容
- **这如何解决消失梯度?**
  - 与LSTM类似，GRU使长期保存信息变得更容易(例如，将update gate设置为0)

LSTM vs GRU

- 研究人员提出了许多门控RNN变体，其中LSTM和GRU的应用最为广泛

- 最大的区别是GRU计算速度更快，参数更少
- 没有确凿的证据表明其中一个总是比另一个表现得更好
- LSTM是一个很好的默认选择(特别是当您的数据具有非常长的依赖关系，或者您有很多训练数据时)
- 经验法则：从LSTM开始，但是如果你想要更有效率，就切换到GRU

Is vanishing/exploding gradient just a RNN problem?

- 不！这对于所有的神经结构(包括前馈和卷积)都是一个问题，尤其是对于深度结构
    - 由于链式法则/选择非线性函数，反向传播时梯度可以变得很小很小
    - 因此，较低层次的学习非常缓慢(难以训练)
    - 解决方案：大量新的深层前馈 / 卷积架构，添加更多的直接连接(从而使梯度可以流动)

例如：

- Residual connections 残差连接又名“ResNet”
- 也称为跳转连接
- 默认情况下，标识连接保存信息
- 这使得深层网络更容易训练

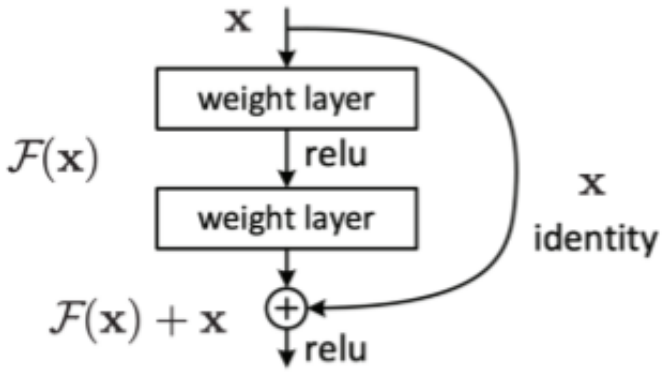


Figure 2. Residual learning: a building block.

例如：

- Dense connections 密集连接又名“DenseNet”
- 直接将所有内容连接到所有内容

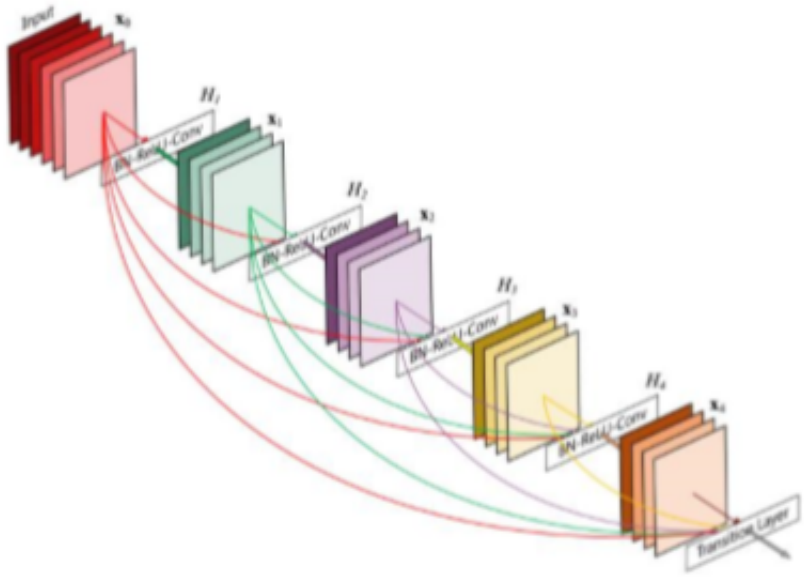


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

例如：

- Highway connections 高速公路连接又称“高速公路网”
- 类似于剩余连接，但标识连接与转换层由动态门控制
- 灵感来自LSTMs，但适用于深度前馈/卷积网络

结论： 虽然消失/爆炸梯度是一个普遍的问题，但由于重复乘以相同的权矩阵，RNN尤其不稳定[Bengio et al, 1994]

Recap

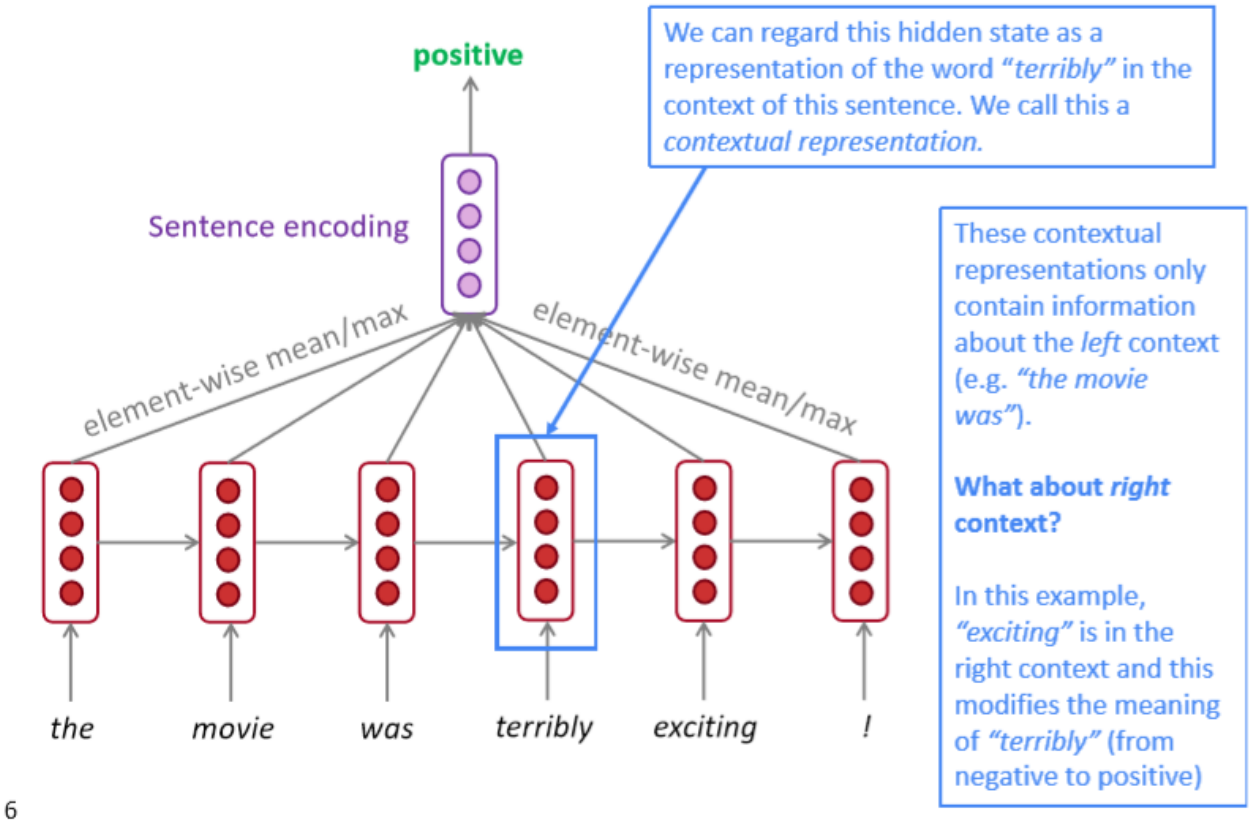
- Today we've learnt:
  - Vanishing gradient problem: what it is, why it happens, and why it's bad for RNNs
  - LSTMs and GRUs: more complicated RNNs that use gates to control information flow; they are more resilient to vanishing gradients

- Remainder of this lecture:
  - Bidirectional RNNs
  - Multi-layer RNNs

  } Both of these are pretty simple

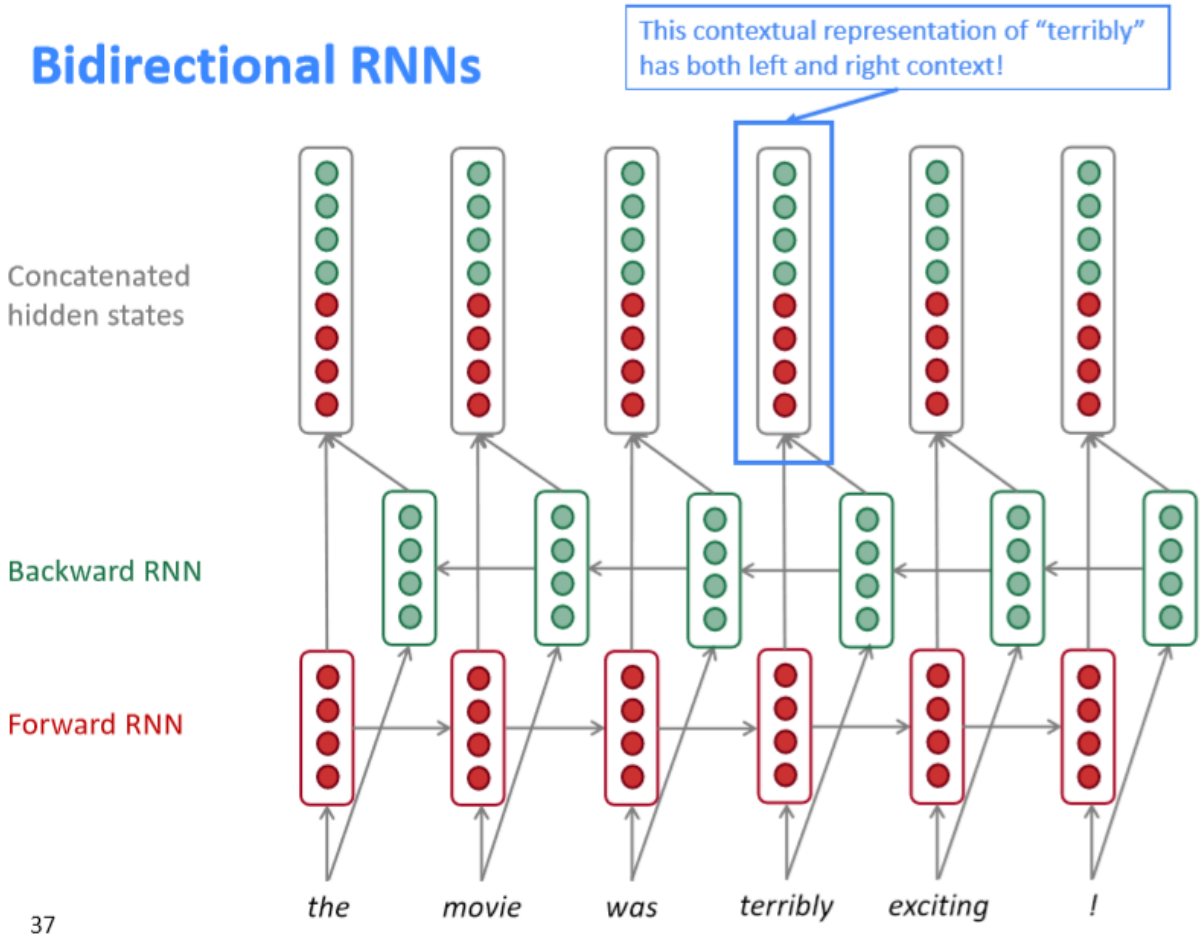
Bidirectional RNNs: motivation

Task: Sentiment Classification



6

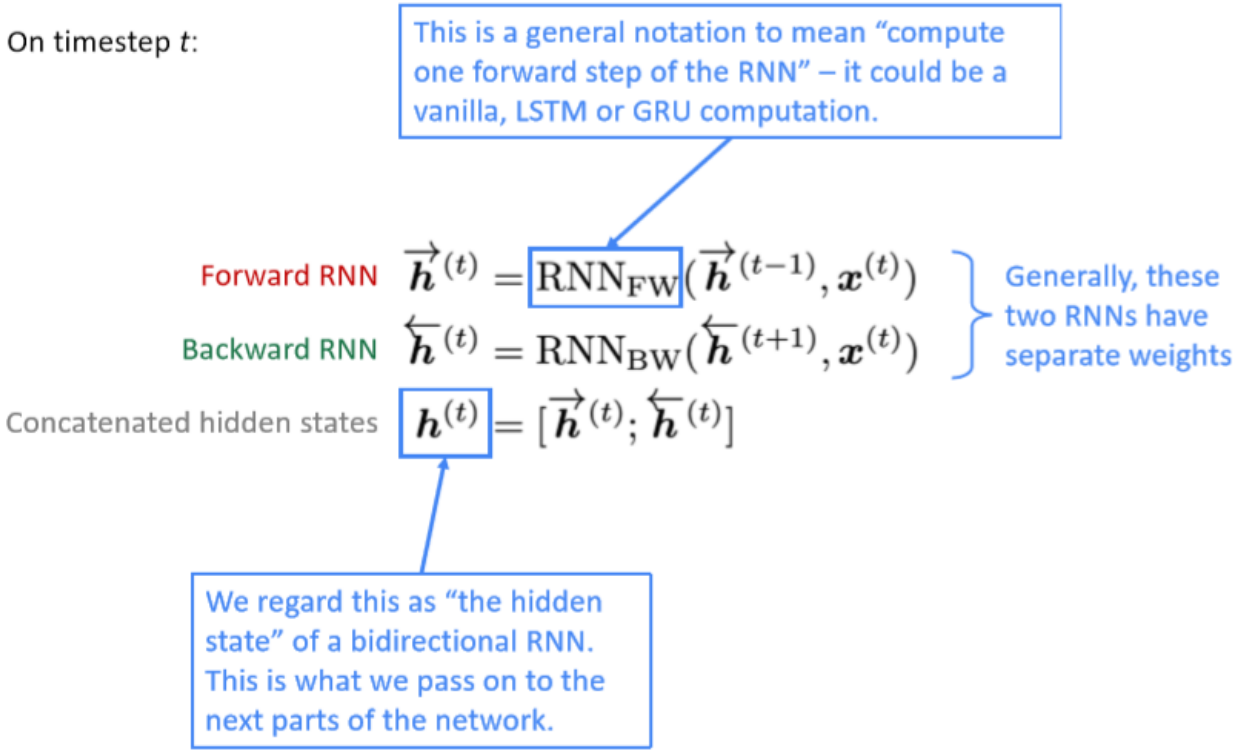
- 我们可以把这种隐藏状态看作是这个句子中单词"terribly"的一种表示。我们称之为上下文表示。
- 这些上下文表示只包含关于左上下文的信息(例如"the movie was")。
- **那么正确的上下文呢?**
- 在这个例子中，"exciting"在右上下文中，它修饰了"terribly"的意思(从否定变为肯定)



37

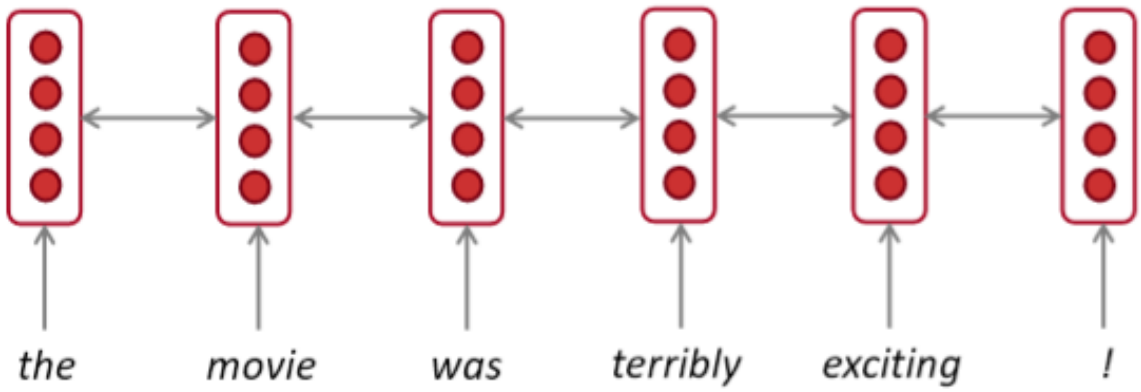
- “terribly”的上下文表示同时具有左上下文和右上下文

Bidirectional RNNs



- 这是一个表示“计算RNN的一个向前步骤”的通用符号——它可以是普通的、LSTM或GRU计算。
- 我们认为这是一个双向RNN的“隐藏状态”。这就是我们传递给网络下一部分的东西。
- 一般来说，这两个RNNs有各自的权重

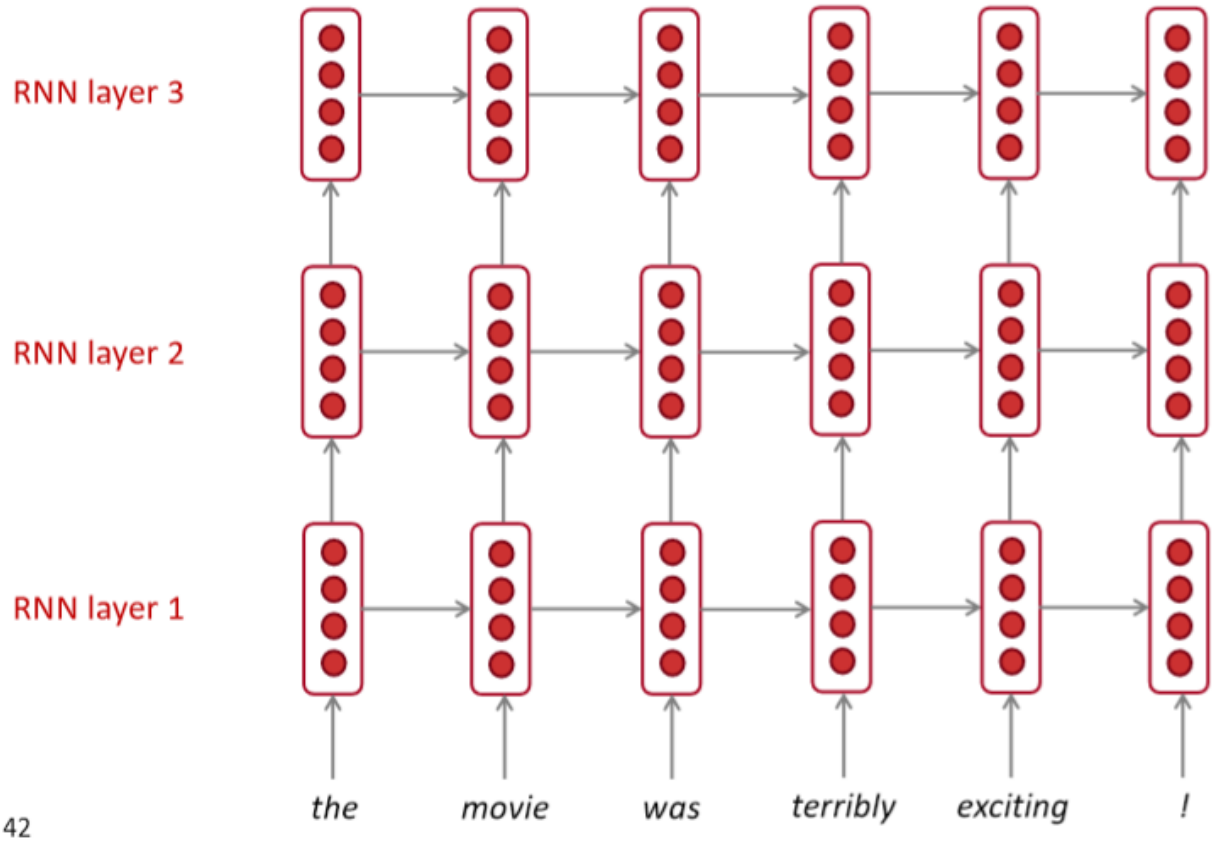
Bidirectional RNNs: simplified diagram



- 双向箭头表示双向性，所描述的隐藏状态是正向+反向状态的连接
- 注意：双向RNNs只适用于访问整个输入序列的情况
  - 它们不适用于语言建模，因为在LM中，您只剩下可用的上下文
- 如果你有完整的输入序列(例如任何一种编码)，双向性是强大的(默认情况下你应该使用它)
- 例如，BERT(来自transformer的双向编码器表示)是一个基于双向性的强大的预训练的上下文表示系统
  - 你会在课程的后面学到更多关于BERT的知识!

Multi-layer RNNs

- RNNs在一个维度上已经是“深的”(它们展开到许多时间步长)
- 我们还可以通过应用多个RNNs使它们“深入”到另一个维度——这是一个多层RNN
  - 较低的RNNs应该计算较低级别的特性，而较高的RNNs应该计算较高级别的特性
- 多层RNNs也称为堆叠RNNs。



42

RNN层 i 的隐藏状态是RNN层 $i+1$ 的输入

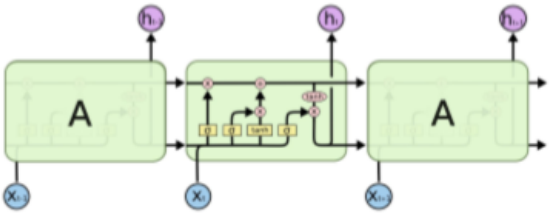
Multi-layer RNNs in practice

- 高性能的RNNs通常是多层的(但没有卷积或前馈网络那么深)
- 例如：在2017年的一篇论文，Britz et al 发现在神经机器翻译中，2到4层RNN编码器是最好的,和4层RNN解码器
  - **Keyphrases**:

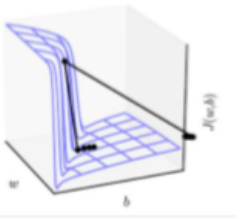
    RNN无法并行化，计算代价过大，所以不会过深
- Transformer-based 的网络(如BERT)可以多达24层。他们有很多skipping-like的连接

In summary

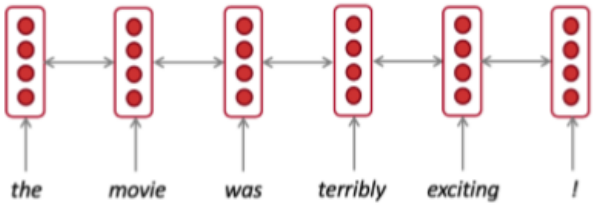
Lots of new information today! What are the practical takeaways?



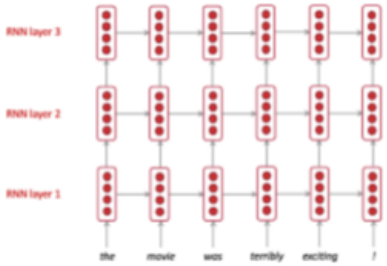
1. LSTMs are powerful but GRUs are faster



2. Clip your gradients



3. Use bidirectionality when possible



4. Multi-layer RNNs are powerful, but you might need skip/dense-connections if it's deep

44

Reference

以下是学习本课程时的可用参考书籍：

《基于深度学习的自然语言处理》（车万翔老师等翻译）

《神经网络与深度学习》

以下是整理笔记的过程中参考的博客：

斯坦福CS224N深度学习自然语言处理2019冬学习笔记目录 (课件核心内容的提炼，并包含作者的见解与建议)

斯坦福大学 CS224n自然语言处理与深度学习笔记汇总 {>>这是针对note部分的翻译<<}