Recurrent Neural Networks - Aug 1'st, 2018

Why sequence models?

Ex. 在现实世界里面，存在很多类型数据都是序列化数据，比如声音、图像、视频、DNA 序列等等，主要有以下应用对序列化数据进行建模.

- Speech recognition(语言识别) $\longrightarrow$ Given an input audio clip X and asked to map it to a text transcript Y.
  - Both the input and the output are sequence data.
  - The input X is an audio clip that plays out over time and Y, the output, is a sequence of words.
  - Example. 开会时，实时记录领导们的讲话，自动字幕生成等等
- Music generation(音乐生成器)
  - Only the output Y is a sequence, the input can be the empty set, or it can be a single integer.
- Sentiment Classification(情感分类)
  - The input X is a sequence, so given the input phrase like, "There is nothing to like in this movie" how many stars do you think this review will be?
- DNA Sequence Analysis（DNA 序列分析）
  - DNA is represented via the four alphabets A, C, G, and T. And so given a DNA sequence can you label which part of this DNA sequence say corresponds to a protein.
- Machine Translation（机器翻译）
  - Given an input sentence, voulez-vou chante avec moi? And you're asked to output the translation in a different language.
- Video Activity Recognition（视频运动识别）
  - Given a sequence of video frames and asked to recognize the activity.
- Name Entity(人名识别)
  - Given a sentence and asked to identify the people in that sentence.

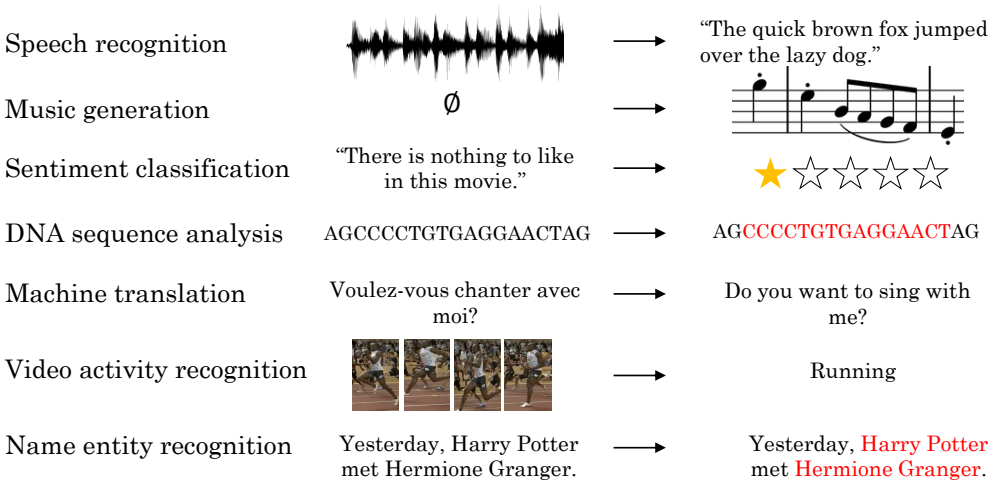


图 1：Examples of sequence data

Notation

下面这句是出自于 J • K • 罗琳写的系列小说哈利 • 波特，现在让序列模型自动识别出句子中哪些单词是人名.

```
Harry Potter and Hermione Granger invented a new spell.
```

这里主要常用在搜索引擎中，比如，过去 24 小时的新闻报道中提到的人名进行恰当的索引，除此以外, 识别系统也能通过名称查找信息，比如人名, 企业名称，时间, 地点, 国家名称货币名称等等.

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

y: 1 1 0 1 1 0 0 0 0.

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<9>}$

$T_x = T_y = 9$

$x^{(i)}$表示第i个样本，$x^{(i)<j>}$表示第i个样本的第j个元素

图 2：Notation

Note: 循环神经网络用于构建从 X 到 Y 的映射

Representing words

使用 one-hot（one-hot 的意思指只有一位为 1 其余位全是 0.）技术对单词进行表示，表示如下图：

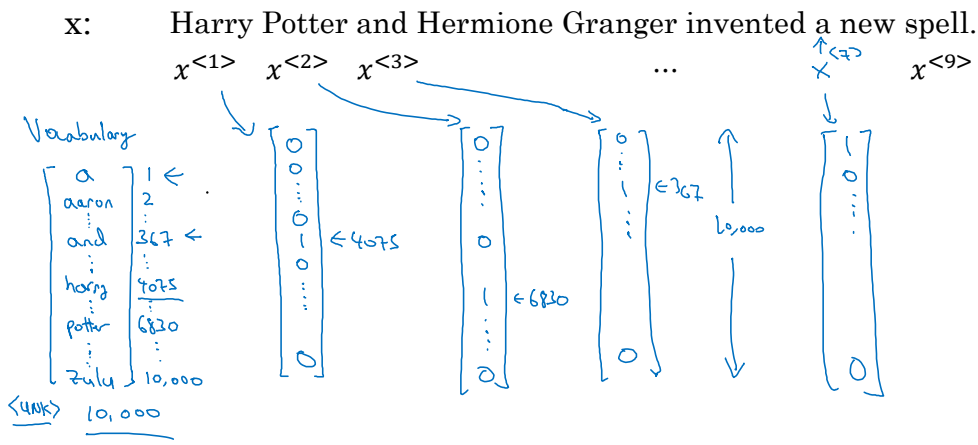


图 3：单词的表示

Note: 当你遇到不再词汇表中的单词怎么办？解决办法是, 创建一个新的标记, 或者说是一个伪词, 称其为 Unknown Word, 并使用尖括号加 UNK(<UNK>)

Recurrent Neural Network Model

传统的神经网络为什么不行？

- 在不同的例子中，输入和输出是不一样的
  - 需要补充每个输入项，使其达到最大长度
- 传统的神经网路结构不会将从不同文本位置上学习到的特征参数进行共享

– 比如在 $x^{<1>}$ 识别到的单词为人名，同样在 $x^{<6>}$ 上识别出来的同样单词也会被识别为人名
– 与 CNN 的权值共享类似

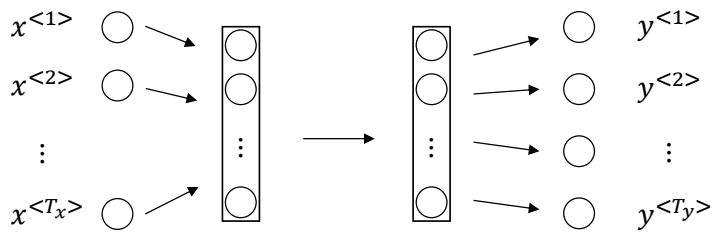


图 4：传统的神经网络，用于序列数据

Recurrent Neural Networks

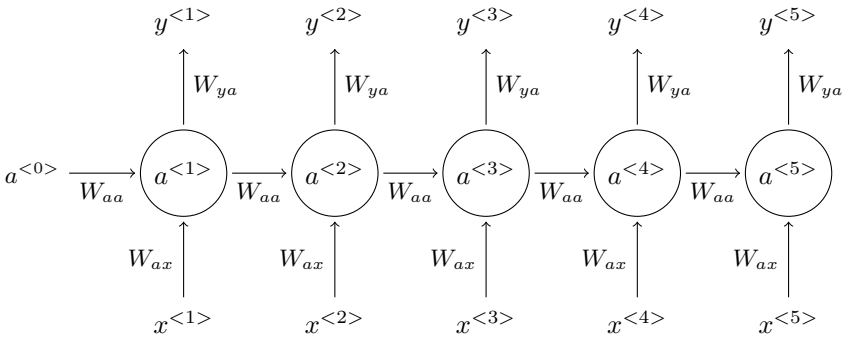


图 5：典型的循环神经网络

其中 $a^{<0>}$ 初始化可以是随机生成或者全零. 但是传统的单向的循环神经网络有以下缺点

- 缺点：仅使用序列之前的部分进行训练
  - Example. He said, "*Teddy* Roosevelt was a great President."
- 改进：使用 BRNN（双向循环神经网络）

Forward Propagation

$$a^{<1>} = g_1(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a) \tag{1}$$
$$\hat{y}^{<1>} = g_2(W_{ay}a^{<1>} + b_y) \tag{2}$$
$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{3}$$
$$\hat{y}^{<t>} = g_2(W_{ay}a^{<t>} + b_y) \tag{4}$$

Simplified RNN notation

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \longrightarrow a^{<t>} = g(\mathbf{W_a}[a^{<t-1>}, x^{<t>}] + b_a)$$

其中 $\mathbf{W_a} = [W_{aa}, W_{ax}]$

Backpropagation through time

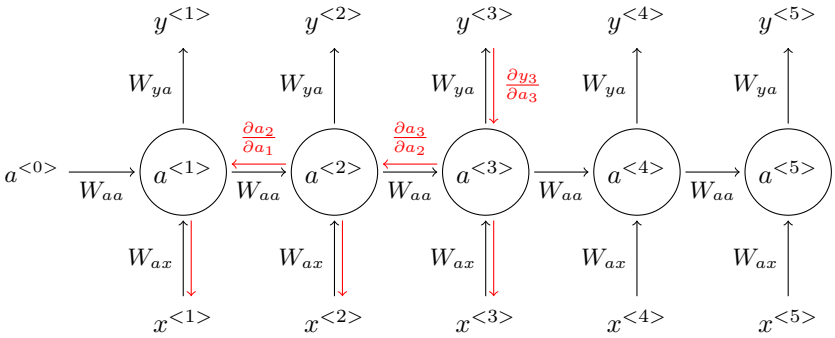


图 6：Backpropagation through time

$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \log (1 - \hat{y}^{<t>})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Different types of RNNs

- One-to-one
- One-to-many
- Many-to-one
- Many-to-many
  - $T_x = T_y$
  - $T_x \neq T_y$

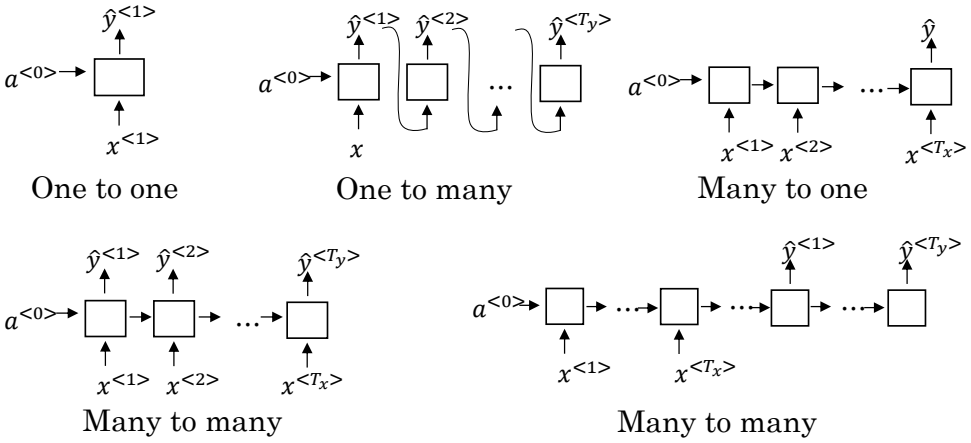


图 7：Different types of RNNs

Language model and sequence generation

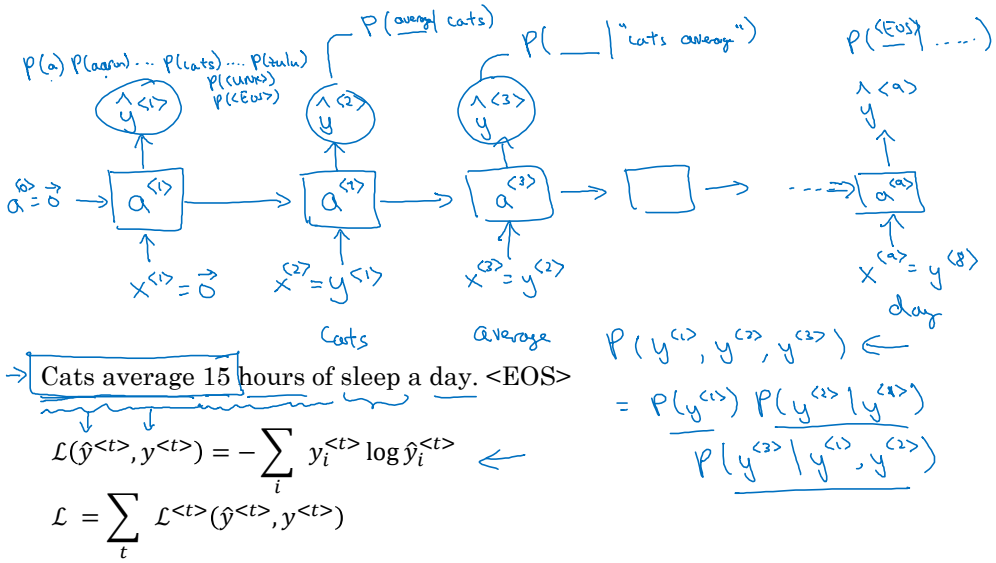


图 8：Language model and sequence generation

Sampling a sequence from a trained RNN

在我们训练了一个序列模型以后，我们可以通过采样新的序列非正式地了解它学到了什么.

一个序列模型模拟 $\longrightarrow$ 在任何序列模型中的词出现的机会如下:

首先为你希望模型生成的第一个字 (word) 采样, 为此你输入 $x^{<1>} = 0\ a^{<0>} = 0$, 现在你的第一个时间标记有一些可能输出的最大概率, 然后根据 softmax 的分布随机采样, softmax 分布会告诉你第一个词指向 a 的概率，指向 Aaron 的概率，指向 Zulu 的概率，第一个单词是未知单词标记的几率是什么. 也许是一个句子标记的结束. 然后你用这个向量使其比如运行 numpy 指令 `np.random.choice` 以采样, 通过这个向量概率定义的分布，然后你就可以为这些**首先出现的词** ($y^{<1>}$) 采样了, 然后你继续，前进到第二步时间点, 记住第二步时间需要输入是这个 $y^{<1>}$, 现在你用刚刚采样过的 $y^{<1>}$, 在这里传递它到下一步时间点, 不管什么方式，管用就行, 你只是选择第一步，把这个输入传递到第二个位置, 然后 softwax 会预测 $y^{<2>}$, 例如, 假设在你对第一个单词进行取样后, 第一个词恰好是 *the*, 这个词作为第一个词很常见. 然后你传递 *the* 作为 $x^{<2>}$, $x^{<2>}$ 现在等于 $y^{<1>}$. 现在你正在试图找出当第一个词是 d 时，第二词的几率, 会是 $y^{<2>}$, 然后你用同样的采样方法为 $y^{<2>}$ 采样, 然后在下一个时间标记, 你把你有的选择当成硬编码, 然后传给下一个时间步骤, 然后你为第三个词取样，不管你选了什么, 你一直继续直到最后一个时间标记, 那你怎么知道序列结束了呢那么, 你可以一直采样直到你取得 EOS 标记.

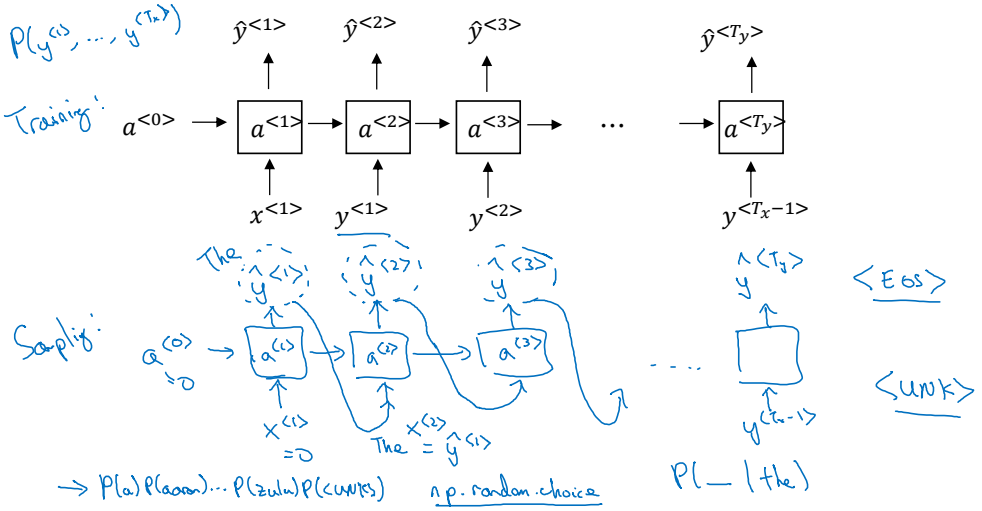


图 9：Sampling a sequence from a trained RNN

字符级别 RNN

在这种情况下, 你的词汇将只是字母. 从 a 到 z, 也许还有空格，标点符号, 如果你需要的话, 数字 0 到 9. 如果你想区分大写和小写, 也可以包括大写字母以及你可以看看你的训练集和里面的字符, 并使用它来定义词汇表. 如果你建立字符级语言模型而不是词汇级的那么你的序列 $y^{<1>}$, $y^{<2>}$, $y^{<3>}$, 将是您的训练数据中的单个字符, 而不是训练数据中的个别单词. 像我们之前的例子一样, 句子“Cats average 15 hours of sleep a day.”, 在这个例子中, c 会是 $y^{<1>}$, a 会是 $y^{<2>}$, t 会是 $y^{<3>}$, 空格会是 $y^{<4>}$ 等等.

使用字符级语言模型有以下利弊：

- 利：不必担心未知的单词标记（<UNK>）
  - 字符级语言模型能够分配一个像 mau 一样的非零概率的序列.
- 缺点：最终会有更长的序列

Note: 字符语言模型不如词级语言模型好

RNNs 中的梯度消失与梯度爆炸

- 原因：语言有长期依赖性
  - Example: *the cat which already ..was full.* vs *the cats which already ..were full.*
  - 解释：句子前面的部分可能会影响句子后面的部分
  - 事实：基本的 RNN 不擅长捕捉长期的依赖关系
- 深度神经网络的梯度消失的问题
  - 在非常深的神经网络情况下 100 层甚至更深，你执行前向传播过程，从左到右，然后反向传播. 当时我们假设，如果这是一个非常深的神经网络那么，从输出 y 得到的梯度很难反向传播去影响前期层的权重，很难影响前期层的计算.
- RNNs 中的梯度消失问题
  - 前向传播，从左到右，然后反向传播，从右到左. 这可能是相当困难的，由于类似的梯度消失问题，由于错误输出关系到之后的步骤，这影响前期的计算. 所以在实践中，这代表很难去让一个神经网络意识到，它需要记住看到了一个单数名词还是复数名词，所以，随后在序列中决定生成“was”还是“were”这取决于它是单数还是复数.
  - 解释：基本 RNN 模型有附近效应，RNN 往往不擅长捕获远程依赖关系.
  - RNN 的缺点.(后面有改进 $\longrightarrow$GRU,LSTM...)
- 深度神经网络的梯度爆炸问题
  - 做反向传播过程，梯度不只是指数递减, 它们也可能随着经历的层数指数增加.
- 事实证明, 梯度消失往往是训练 RNNs 时更严重的问题, 尽管发生梯度爆炸时, 这可能是灾难性的, 因为指数级大的梯度可能导致参数变得很大，以至于你的神经网络参数变得非常混乱. 事实证明, 爆炸梯度更容易被发现, 因为参数只是爆炸的话, 你可能会经常看到 NaN 或者非数字, 这意味着在你的神经网络计算中数值溢出的结果.
- 梯度爆炸解决方法：gradient clipping
  - 如果它大于某个临界值重新缩放某些梯度向量, 使其不那么大.

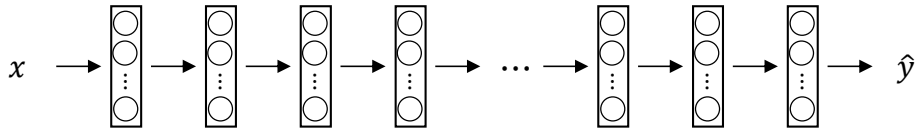
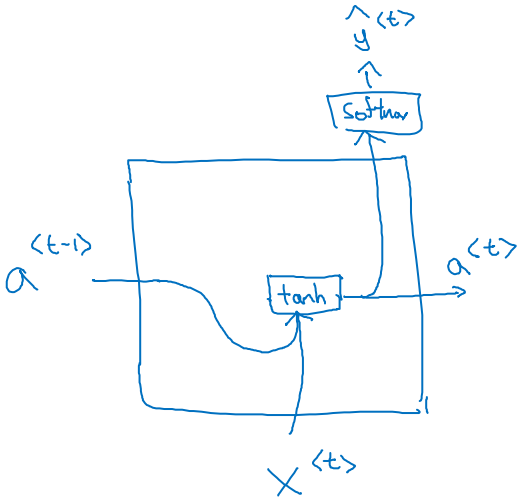


图 10：深度神经网络

GRU-允许神经网络来捕获更长范围的依赖性

- GRU ⟶ greater recurrent units(门控循环单元)
  - 解决梯度消失问题的一个有效解决方案
  - 修改了基本 RNN 的隐含层，使它更好的捕捉到更长范围的依赖性，帮助解决梯度消失问题.
  - 允许神经网络来捕获更长范围的依赖性



$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

图 11：Basic RNNs Cell

$$\hat{c}^{<t>} = tanh\ (W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c) \tag{5}$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u) \tag{6}$$

$$\Gamma_r = \sigma\ (W_r[c^{<t-1>}, x^{<t>}] + b_r) \tag{7}$$

$$c^{<t>} = \Gamma_u\ * \hat{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>} \tag{8}$$

$$a^{<t>} = c^{<t>} \tag{9}$$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}} \tag{10}$$

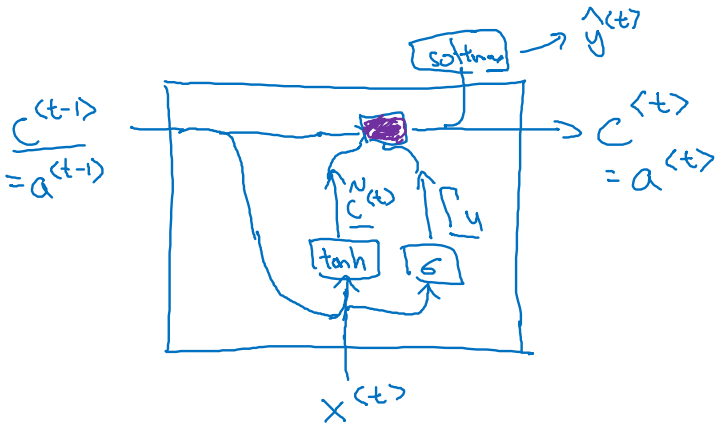


图 12：GRU Cell,Γ_u 为更新门，Γ_r 为相关门

LSTM

- LSTM $\longrightarrow$ 长短期记忆单元

$$\hat{c}^{<t>} = tanh\ (W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c) \tag{11}$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u) \tag{12}$$

$$\Gamma_f = \sigma(W_f[c^{<t-1>}, x^{<t>}] + b_f) \tag{13}$$

$$\Gamma_o = \sigma\ (W_o[c^{<t-1>}, x^{<t>}] + b_o) \tag{14}$$

$$c^{<t>} = \Gamma_u\ * \hat{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{15}$$

$$a^{<t>} = \Gamma_o * c^{<t>} \tag{16}$$

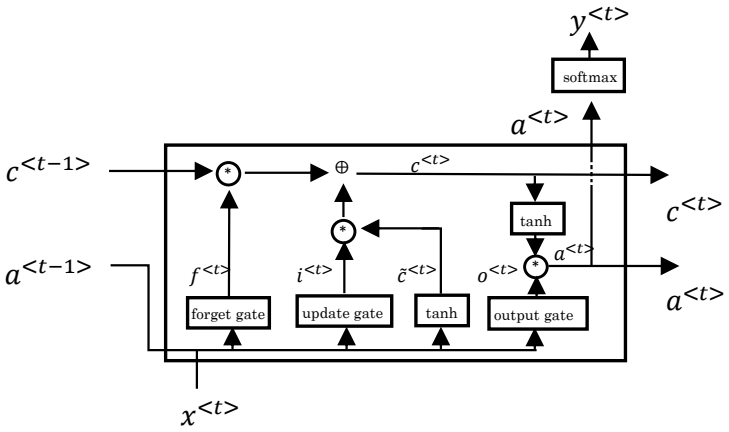


图 13：LSTM Cell

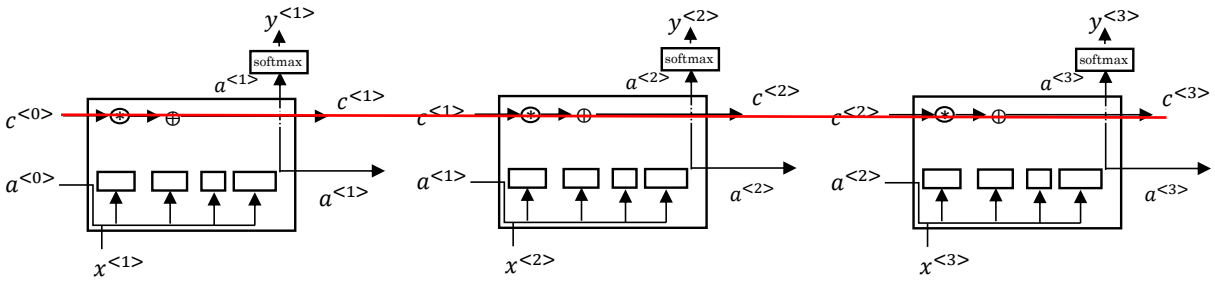


图 14：Multi-LSTM Cell Connection

GRU vs. LSTM

- GRU
  - 模型简单，适合构建较大的网络；
  - 只有两个门控，从计算角度看，效率更高；
  - 可扩展性更强；
  - 近些年，GRU 的势头越来越猛.
- LSTM
  - 更加的强大和灵活；
  - 三个门控；
  - 经过历史检验的方法, 提出的时间早于 GRU.

BRNN

双向 RNN 的工作原理如下：我将用一个简化的四维输入, 或者说是一个四个单词的句子. 所以我们有四个输入矢量.$x^{<1>}$ 到 $x^{<4>}$ 因此, 这个网络的头部层将有一个前向递归的部分. 然后我会画一个右箭头去表示这是前向递归的组成, 因此, 它们将连接如下图，因此, 这四个单位每次都输入当前的 X, 然后喂入数据去帮助预测 $y^{<1>}$, $y^{<2>}$, $y^{<3>}$ 和 $y^{<4>}$，有 $\overleftarrow{a^{<1>}}$, 左箭头表示这是一个后向连接, $\overleftarrow{a^{<2>}}$，向后，$\overleftarrow{a^{<3>}}$, 向后，$\overleftarrow{a^{<4>}}$，向后, 因此左箭头表示它是后向连接的, 因此, 我们将网络连接成如下. 而这后向的连接将连接彼此，向后传播.

此网络定义了一个无循环图. 因此, 给定一个输入序列, $x^{<1>}$ 到 $x^{<4>}$, 前向序列将首先计算 $a^{<1>}$，然后用它计算 $\vec{a}^{<2>}$, 然后 $\vec{a}^{<3>}$,$\vec{a}^{<4>}$, 相反, 向后序列将从计算 $\overleftarrow{a^{<4>}}$ 开始, 然后向后计算 $\overleftarrow{a^{<3>}}$, 然后当计算到激活层时, 注意这整个过程不是反向传播, 而是前向传播. 前向传播的关系图中有一部分计算是从左向右，有一部分是从右向左. 计算了 $\overleftarrow{a^{<3>}}$ 以后, 您可以使用这些激活部分来计算 $\overleftarrow{a^{<2>}}$, 然后 $\overleftarrow{a^{<1>}}$, 然后计算所有的激活部分, 然后你就可以做出预测了.

考虑过去和未来的信息 $\longrightarrow$ 双向递归神经网络（BRNN）

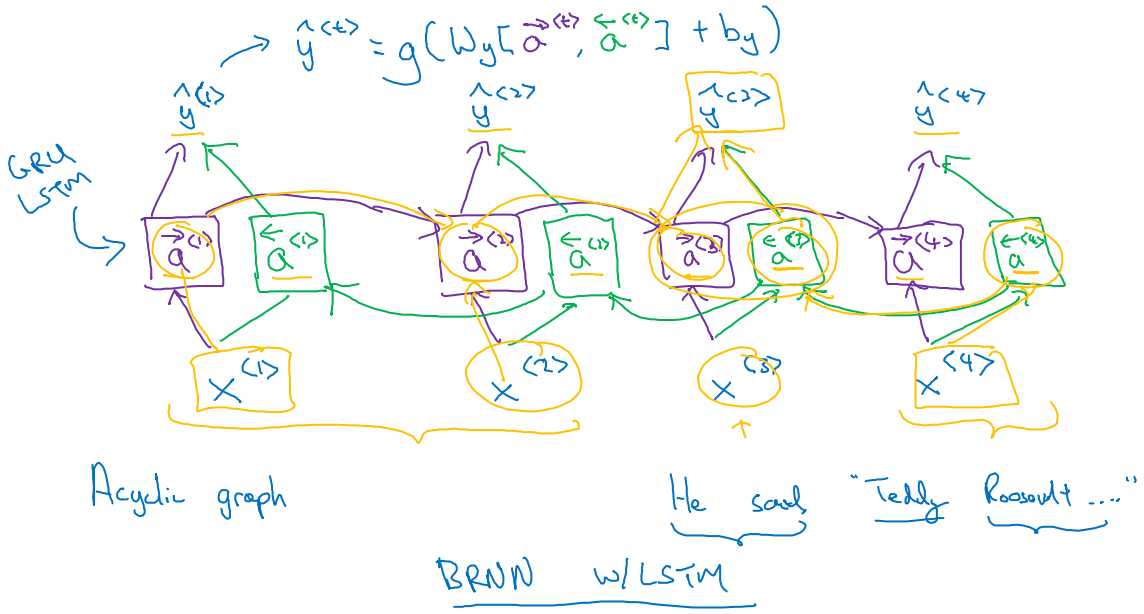


图 15：BRNN

Deep RNNs

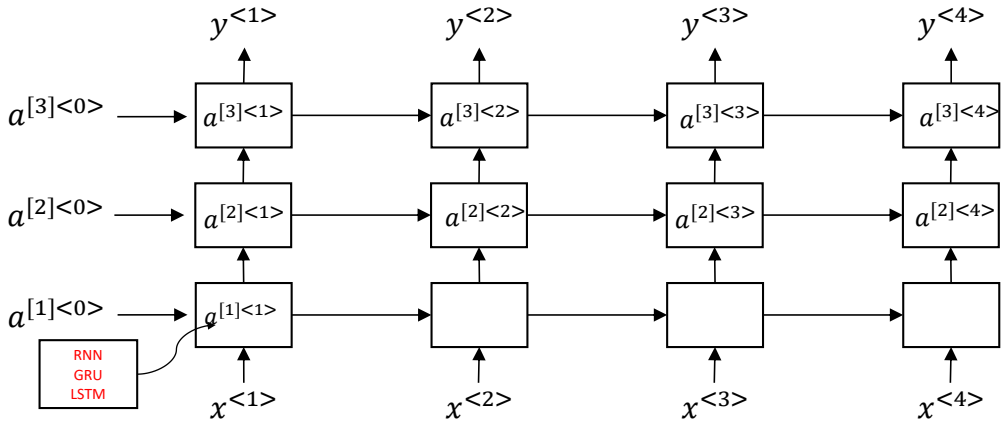


图 16：Deep RNNs