1、序列模型简介

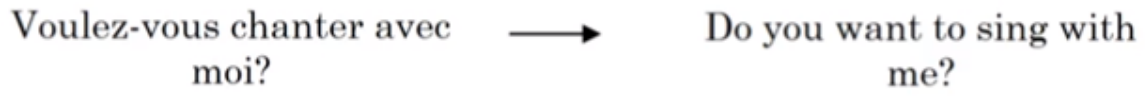
不同于之前的图像识别，序列模型的输入或输出是一段连续的值，譬如：

语音识别

语句情感识别

基因序列分析

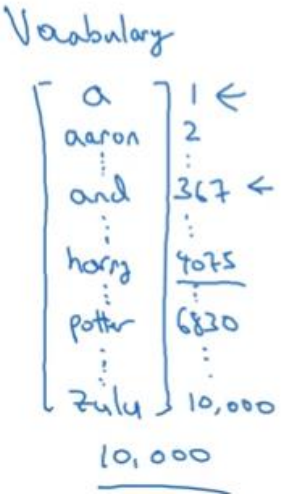
文字翻译



视频分析

2、NLP 初识

在序列模型中，语句识别是一个典型的案例，通常在做自然语言处理（Natural Language Processing）的时候，需要构建一个词典，形如：

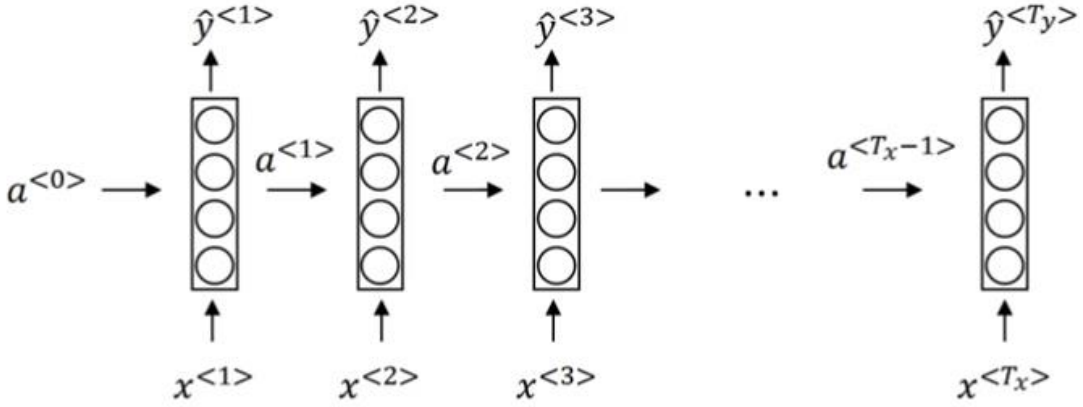


一般的商业用词典可能达到百万条记录，此次实验用 1W 条记录，针对每一个训练语句，将单词单独提取出来，进行独热编码处理，这样一条语句就有多个向量特征了，接下来要做的就是将输入特征与输出 y 进行映射关联。

3、循环神经网络 RNN

在语句处理的时候，由于不可能一次只把一个单词对应到输出标签上（两个问题：长度不一样，导致每次输出不同；缺

失了单词之间的联系）那么考虑到在进行第 n+1 个单词的预测时，最好能将前 n 个单词的信息也包含进来，所以提出了循环神经网络的概念，先上图：



X1 代表第一个单词的输入特征，通过一层或多层计算得到一个输出值，在进行第二个单词的计算的时候，将第一个单词的信息值输入其中，这样就相当于神经网络里构成一个回路，需特别说明的是，第一次输入的时候，会给一个特殊的值。这样一来，就避免了第一层要输入全部的特征值而导致参数值很多的问题。

个人理解 RNN 模型就是一个普通神经网络的不同对象的组合，在之前的神经网络学习中，每个对象都是单独分离的，神经网络对一个对象的输出不会影响别的对象，而在 RNN 中，一个对象的输出是作为下一个对象计算时的输入，相当于改变了下一个对象原有的特征，这便是 RNN，与普通网络类似，它也有梯度爆炸、消失问题。

4、前向传播

由上方这副图，我们可以得到前向传播的公式。

$$a^{<t>} = g(W_{aa}a^{<t>} + W_{ax}x^{<t>} + b_a)$$

该公式是用来计算循环点处的输出计算，可以稍加变换得到，其中，

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

Wa = [Waa：Wax]，这样可以精简很多。

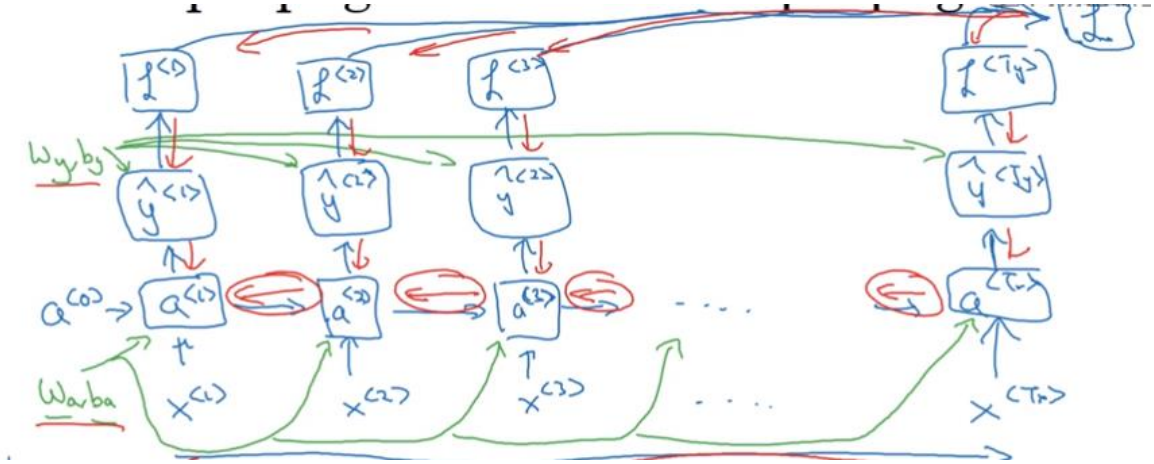
而下面的公式是关于 y 的输出，即真正要进行输出的时候进行的。

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

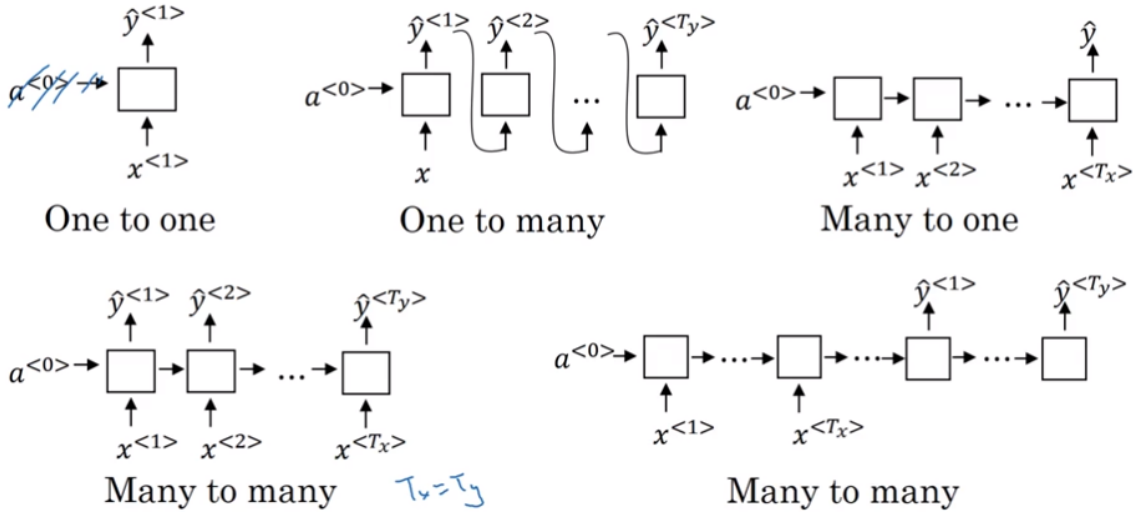
5、反向传播

首先定义损失函数，因为每一个单词有一个输出 y 标签，所以 t 个单词就有 t 个输出标签，用交叉熵损失函数（cross entropy）（就像 logistic 回归的损失函数那样）将每个

损失值加起来，得到总的损失函数，再进行反向传播，其反向过程流如下，这样一步一步优化权重矩阵 Waa，因为这样的计算方式，所以越浅层的矩阵，优化的次数更多。



6、序列模型



之前讲文本命名实体问题，属于多对多问题，且输入等于输出，除了该类型，还有，一对一，一对多（音乐生成），多对一（情感分类），多对多（命名实体，文字翻译）

7、训练序列模型

序列模型中最基础的一个网络结构就是一对多模型，也称生成模型，即给定一句话（词）中第一个单词（字母），网络预测接下来可能的单词（字母）。为训练这样的模型，首先需要大量的文本内容作为语料库，然后构成自己的词向量表，作为训练集的特征和输出标签。接下来搭建模型，譬如有这样一句话：

Cats average 15 hours of sleep a day.<EOS>

然后转换为 one-hot 编码，接着搭建模型，在第一个时间步中，我们给出句子第一个正确单词，输出标签设置为第二个单词的 one-hot 码，接着在第二个时间步中，输入第二个正确单词，输出标签设置为第三个单词的 one-hot 码，这样一直持续到倒数第二个单词的输入时间步，最后计算代价函数、梯度优化，这样预测时只需输入第一个单词，就可得到完整的句子。

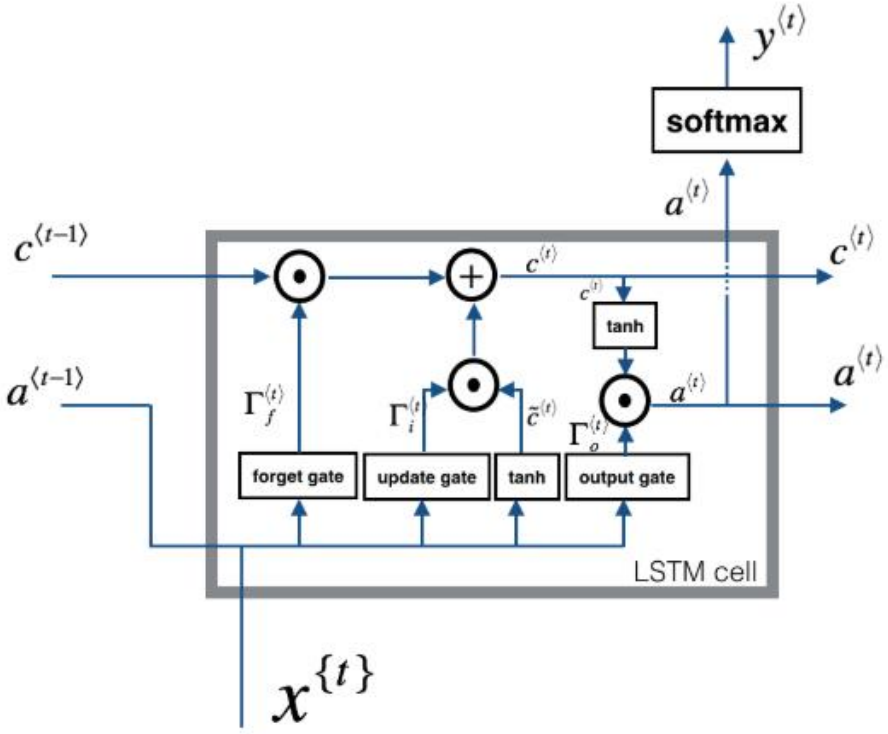
8、梯度消失/爆炸

之前讲了当浅层的权重矩阵小于 1 或者大于 1 时，就会在前向传播的时候出现梯度消失或爆炸问题，同样在反向的梯度计算的时候，也会出现该类问题，对于梯度爆炸问题，容易及时发现，且采用梯度修剪的方法就很容易预防，但对于梯度消失问题就另当别论了，需采用 GRU（门控循环单元网络）、LSTM（长短时记忆）。

对于序列模型，出现梯度消失是一个问题，这会导致前面浅层的单词很难影响到后面深层的单词，比如：The cat eat many food, such as ……, was full. 如果梯度消失的话，则 cat 难以影响到 was 单词上。

9.1、LSTM

LSTM 网络是一种能够帮助解决梯度消失问题的一种方法，通过构造 LSTM 单元来生成 LSTM 网络，其原理是将浅层的内容含义输出至深层，这样一条语句的不同词汇之间的关联长度就会加长，避免了梯度消失问题。下面将介绍 LSTM 单元的构成组件。



上图是一个 LSTM 单元，包含以下组件：

• 遗忘门：用来摆脱无需记忆的内容和留住需要记忆的内容，公式为：

$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$

其值在 0-1 之间，之后与 $c^{\langle t-1 \rangle}$相乘，如果值为 0，意味删除某一信息，为 1 则保留。

• 更新门：遗忘门用来遗忘之前信息，更新门用来产生需要新记住的信息。公式：

$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

另外，需要被更新的信息为 $c\sim\langle t \rangle$，计算公式为：

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

• 更新单元：将遗忘信息和更新信息结合起来：输出记忆状态

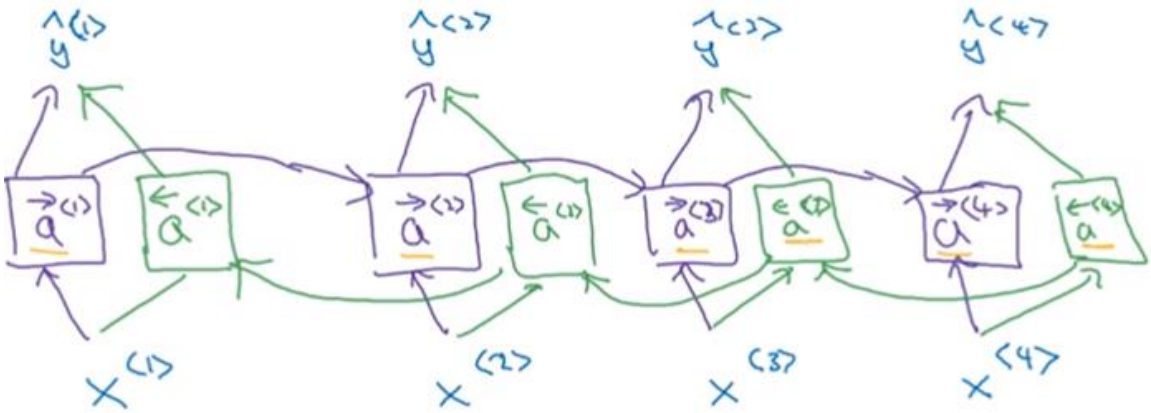
$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} * c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} * \tilde{c}^{\langle t \rangle}$$

• 输出门：输出隐藏状态

$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o) \qquad a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} * \tanh(c^{\langle t \rangle})$$

10、双向的 RNN

传统的 RNN 模型只是简单的”前向”传播，使得浅层单词无法获得深层单词含义，而双向的 RNN 可以解决这个问题，如图是一个双向版本的 RNN，在最后一层获得激活值

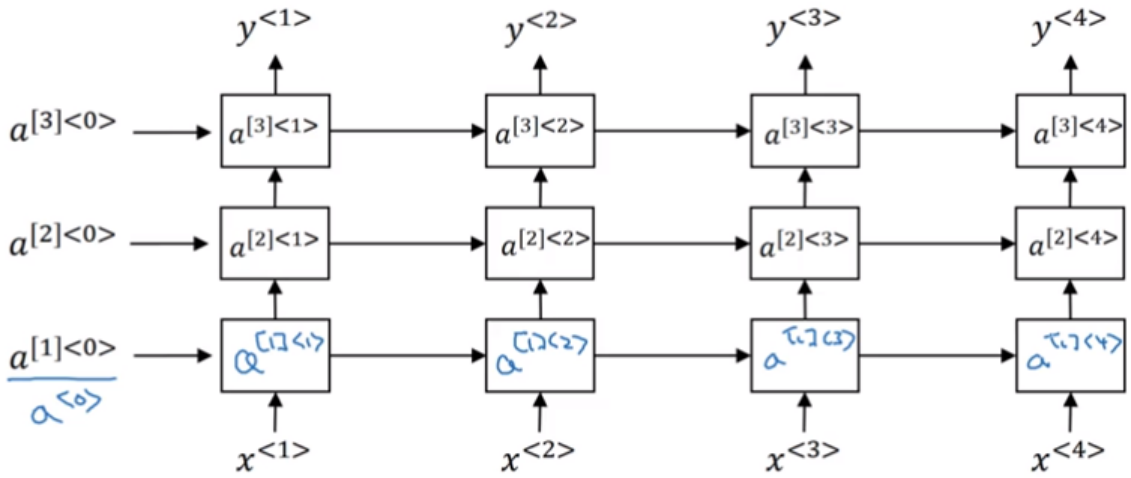


a^4 后，通过”反向”传播，得到 $a^{3'}$、$a^{2'}$、$a^{1'}$，然后输出的时候使用公式：这样就保证了浅层词汇受到了深层词汇的影响。

$$\hat{y}^{\langle t \rangle} = g(W_y[\overrightarrow{a}^{\langle t \rangle}, \overleftarrow{a}^{\langle t \rangle}] + b_y)$$

11、深层 RNN

之前讲的 RNN 模型都是在一个时间线上传递的，然而它也有深层次的版本，比如



此时 $a^{[2]<2>} = g(Wa^{[2]}[a^{[2]<1>}, a^{[1]<2>}] + ba^{[2]})$，在深层 RNN 模型中，3 层已经属于比较深层的模型了，训练算力需求很大。