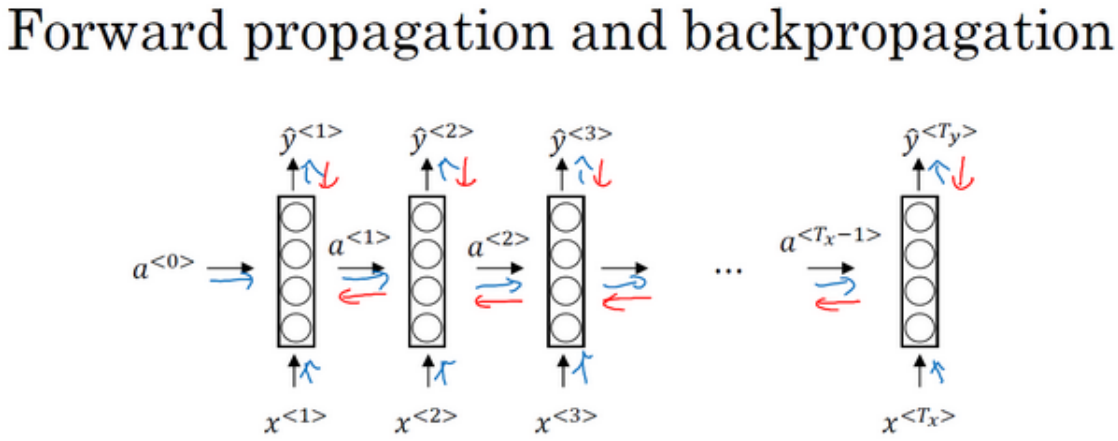
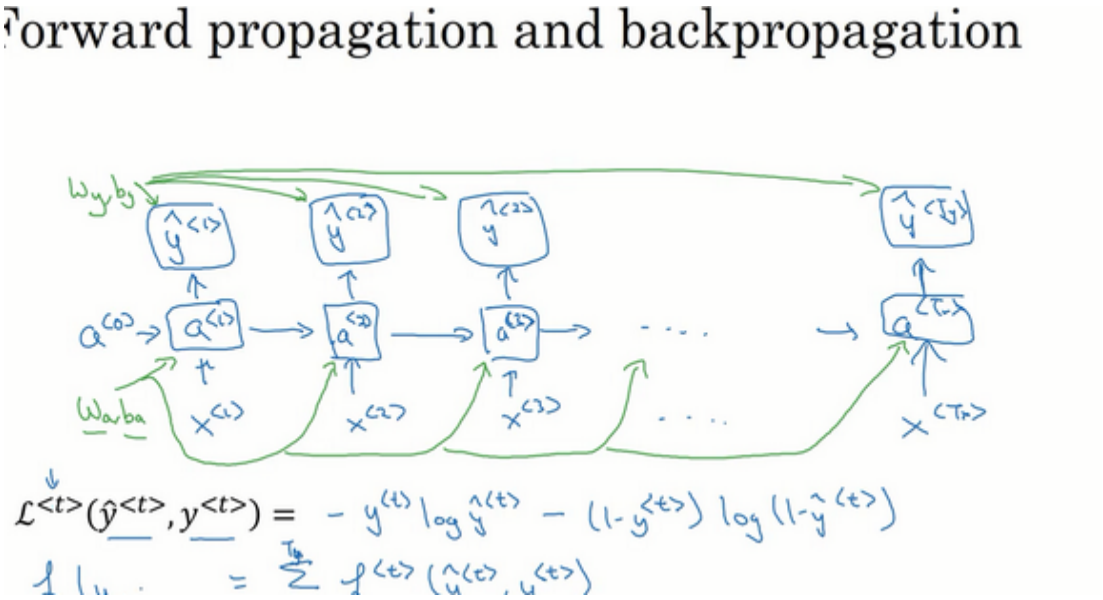
## 1.4 通过时间的反向传播（Backpropagation through time）

之前我们已经学过了循环神经网络的基础结构，在本节视频中我们将来了解反向传播是怎样在循环神经网络中运行的。和之前一样，当你在编程框架中实现循环神经网络时，编程框架通常会自动处理反向传播。但我认为，在循环神经网络中，对反向传播的运行有一个粗略的认识还是非常有用的，让我们来一探究竟。

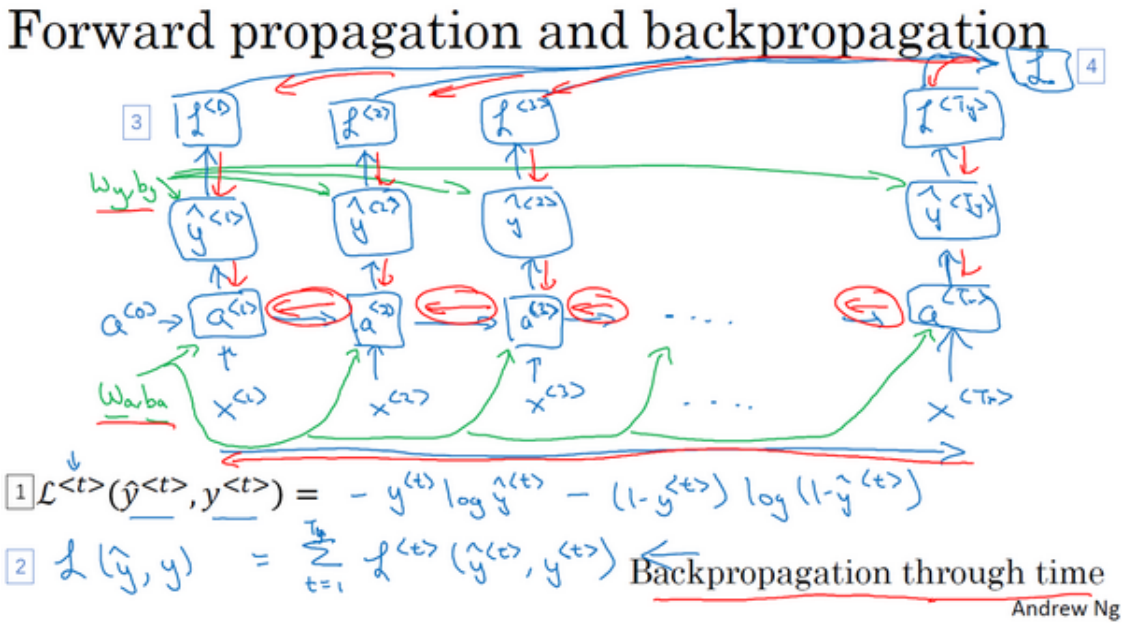



在之前你已经见过对于前向传播（上图蓝色箭头所指方向）怎样在神经网络中从左到右地计算这些激活项，直到输出所有地预测结果。而对于反向传播，我想你已经猜到了，反向传播地计算方向（上图红色箭头所指方向）与前向传播基本上是相反的。




我们来分析一下前向传播的计算，现在你有一个输入序列，$x^{<1>}$，$x^{<2>}$，$x^{<3>}$一直到$x^{<T_x>}$，然后用$x^{<1>}$还有$a^{<0>}$计算出时间步 1 的激活项，再用$x^{<2>}$和$a^{<1>}$计算出$a^{<2>}$，然后计算$a^{<3>}$等等，一直到$a^{<T_x>}$。

为了真正计算出$a^{<1>}$，你还需要一些参数，$W_a$和$b_a$，用它们来计算出$a^{<1>}$。这些参数在之后的每一个时间步都会被用到，于是继续用这些参数计算$a^{<2>}$，$a^{<3>}$等等，所有的这些激活项都要取决于参数$W_a$和$b_a$。有了$a^{<1>}$，神经网络就可以计算第一个预测值$\hat{y}^{<1>}$，接着到下一个时间步，继续计算出$\hat{y}^{<2>}$，$\hat{y}^{<3>}$，等等，一直到$\hat{y}^{<T_y>}$。为了计算出$\hat{y}$，需要参数$W_y$和$b_y$，它们将被用于所有这些节点。




然后为了计算反向传播，你还需要一个损失函数。我们先定义一个元素损失函数（上图编号 1 所示）

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

它对应的是序列中一个具体的词，如果它是某个人的名字，那么$y^{<t>}$的值就是 1，然后神经网络将输出这个词是名字的概率值，比如 0.1。我将它定义为标准逻辑回归损失函数，也叫交叉熵损失函数（Cross Entropy Loss），它和之前我们在二分类问题中看到的公式很像。所以这是关于单个位置上或者说某个时间步$t$上某个单词的预测值的损失函数。

现在我们来定义整个序列的损失函数，将$L$定义为（上图编号 2 所示）

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

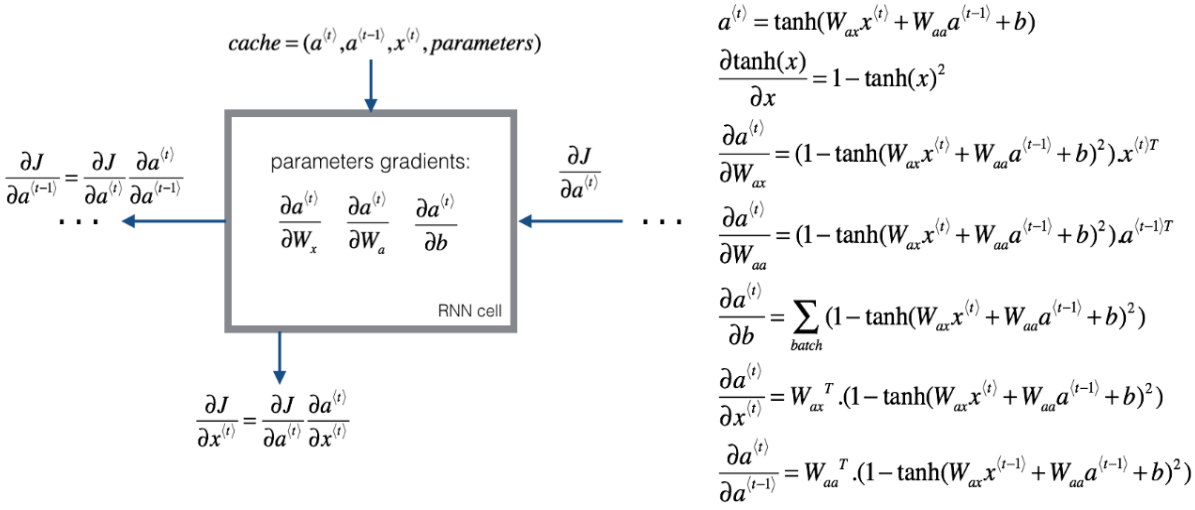
在这个计算图中，通过$\hat{y}^{<1>}$可以计算对应的损失函数，于是计算出第一个时间步的损失函数（上图编号 3 所示），然后计算出第二个时间步的损失函数，然后是第三个时间步，一直到最后一个时间步，最后为了计算出总体损失函数，我们要把它们都加起来，通过下面的等式（上图编号 2 所示的等式）计算出最后的$L$（上图编号 4 所示），也就是把每个单独时间

步的损失函数都加起来。

这就是完整的计算图，在之前的例子中，你已经见过反向传播，所以你应该能够想得到反向传播算法需要在相反的方向上进行计算和传递信息，最终你做的就是把前向传播的箭头都反过来，在这之后你就可以计算出所有合适的量，然后你就可以通过导数相关的参数，用梯度下降法来更新参数。

在这个反向传播的过程中，最重要的信息传递或者说最重要的递归运算就是这个从右到左的运算，这也就是为什么这个算法有一个很别致的名字，叫做**“通过（穿越）时间反向传播（backpropagation through time）”**。取这个名字的原因是对于前向传播，你需要从左到右进行计算，在这个过程中，时刻$t$不断增加。而对于反向传播，你需要从右到左进行计算，就像时间倒流。“通过时间反向传播”，就像穿越时光，这种说法听起来就像是你需要一台时光机来实现这个算法一样。

**RNN** 反向传播示意图：




希望你大致了解了前向和反向传播是如何在 **RNN** 中工作的，到目前为止，你只见到了 **RNN** 中一个主要的例子，其中输入序列的长度和输出序列的长度是一样的。在下节课将展示更多的 **RNN** 架构，这将让你能够处理一些更广泛的应用。