全面理解RNN及其不同架构

原创 Evan AI遇见机器学习 2018-03-09

1. 引出循环神经网络
2. RNN讲解
3. 为什么可以参数共享的简单理解
4. rnn的训练
5. RNN几种架构
6. 引出下文：Long-Term依赖问题

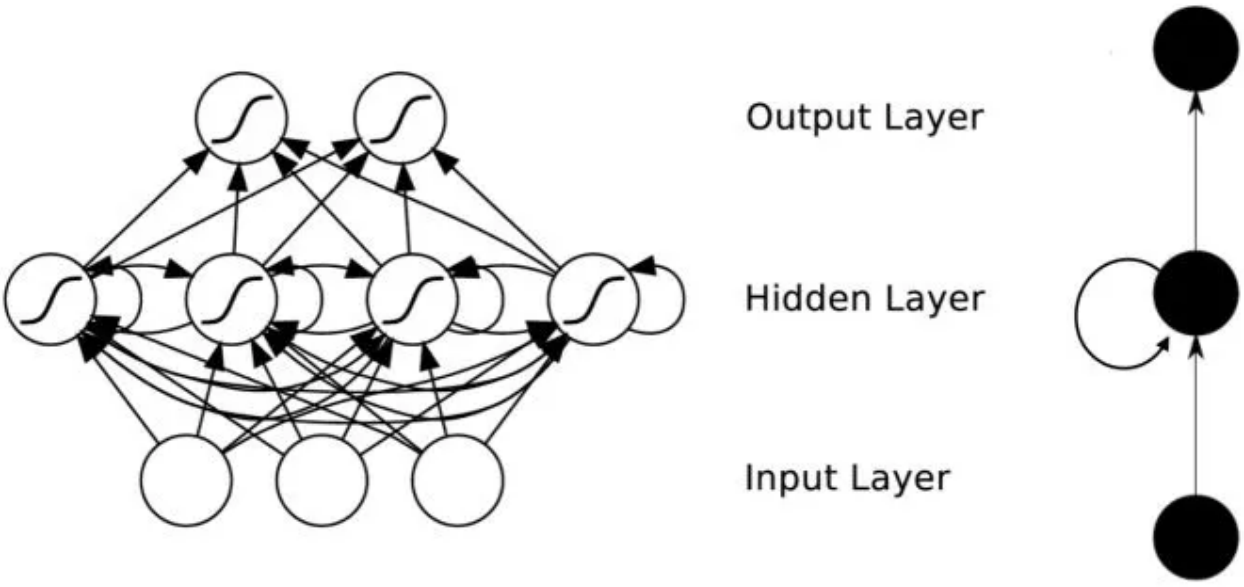
引出循环神经网络

在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的。但是这种普通的神经网络对于很多问题却无能无力。例如时序问题，比如你要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。这个时候，我们怎么办呢？下面有两种解决方案

1. 一种思路是记忆之前的分类器的状态，在这个基础上训练新的分类器，从而结合历史影响，但是这样需要大量历史分类器
2. 重用分类器，只用一个分类器总结状态，其他分类器接受对应时间的训练，然后传递状态,这样就避免了需要大量历史分类器，而且还比较有效的解决了这个问题。而这样一种东西是什么呢？没错，就是RNN(循环神经网络)

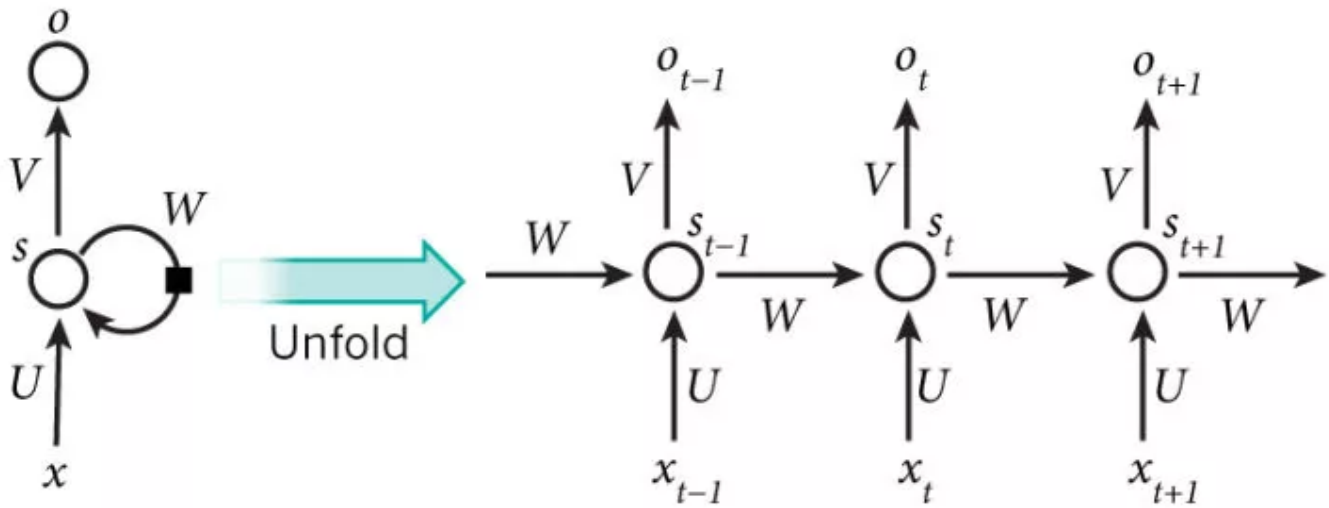
RNN讲解

RNN出现的**目的是来处理序列数据的**。RNN之所以称为循环神经网路，是因为一个序列当前的输出与前面的输出有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，也就是说隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。理论上，RNN能够对任何长度的序列数据进行处理。但是在实践中，为了降低复杂性往往假设当前的状态只与前面的几个状态相关，下图便是一个典型的RNN：



RNN包含输入单元(Input units)，输出单元(Output units)和隐藏单元(Hidden units)。输入为 $\{x_0, x_1, \ldots, x_t, x_{t+1}, \ldots\}$ 的数据，得到隐含层 $\{s_0, s_1, \ldots, s_t, s_{t+1}, \ldots\}$ 和输出层 $\{y_0, y_1, \ldots, y_t, y_{t+1}, \ldots\}$ 的数据。

如上右图，隐藏单元(Hidden units)往往最为主要。你会发现，在图中：有一条单向流动的信息流是从输入单元到达隐藏单元的，与此同时另一条单向流动的信息流从隐藏单元到达输出单元。在某些情况下，RNN会打破后者的限制，引导信息从输出单元返回隐藏单元，这些被称为“Back Projections”，并且隐藏层的输入还包括上一隐藏层的状态，即隐藏层内的节点可以自连也可以互连。



为了更简单的理解，现在我们把循环神经网络进行展开成一个全神经网络。例如，对一个包含3个单词的语句，那么展开的网络便是一个有3层神经网络，每一层代表一个单词。对于该网络的计算过程如下： x_t 表示第 $t, t=1,2,3...$ 步(step)的输入。比如 x_1 为第二个词的词向量(x_0 为第一个词)；

s_t 为隐藏层的第t步的状态，它是网络的记忆单元。 s_t 根据当前输入层的输出与上一步隐藏层的状态进行计算。 $s_t = f(U * x_t + W * s_{t-1})$ ，其中f一般是非线性的激活函数，如tanh或ReLU，在计算 s_0 时，即第一个单词的隐藏层状态，需要用到s−1，但是其并不存在，在实现中一般置为0向量； o_t 是第t步的输出，如下个单词的向量表示， $o_t = softmax(V * s_t)$.

需要注意的是：在RNN中，每输入一步，每一层各自都共享参数U,V,W。其反映着RNN中的每一步都在做相同的事，只是输入不同，因此大大地降低了网络中需要学习的参数，具体原因下面说。还有就是上图中每一步都会有输出，但是每一步都要有输出并不是必须的。比如，我们需要预测一条语句所表达的情绪，我们仅仅需要关系最后一个单词输入后的输出，而不需要知道每个单词输入后的输出。同理，每步都需要输入也不是必须的，其他架构后面也会说。反正时时记得：RNN的关键之处在于隐藏层，隐藏层能够捕捉序列的信息。

> 需要注意的是，那对每一个词做处理的cell来说，他并不是只有1个神经元的，而是n个hidden units，这个在tensorflow中是可以设置的，可以看成是神经网络的宽度！

为什么可以参数共享的简单理解:

对于RNN，我的理解是对于一个句子或者文本，那个参数可以看成是语法结构或者一般规律，而下一个单词的预测必须是上一个单词和一般规律或者语法结构向结合的。我们知道，语法结构和一般规律在语言当中是共享的，所以，参数自然就是共享的！

用通俗的例子解释（ 知乎@YJango大佬 的例子）：

例1：轨迹的产生，如地球的轨迹有两条线索决定，其中一条是地球自转，另一条是地球围绕太阳的公转。下图是太阳和其他星球。自转相当于 $W_{xh} * x_t$ ，而公转相当于 $W_{hh} * h_{t-1}$ ，二者共同决定实际轨迹。

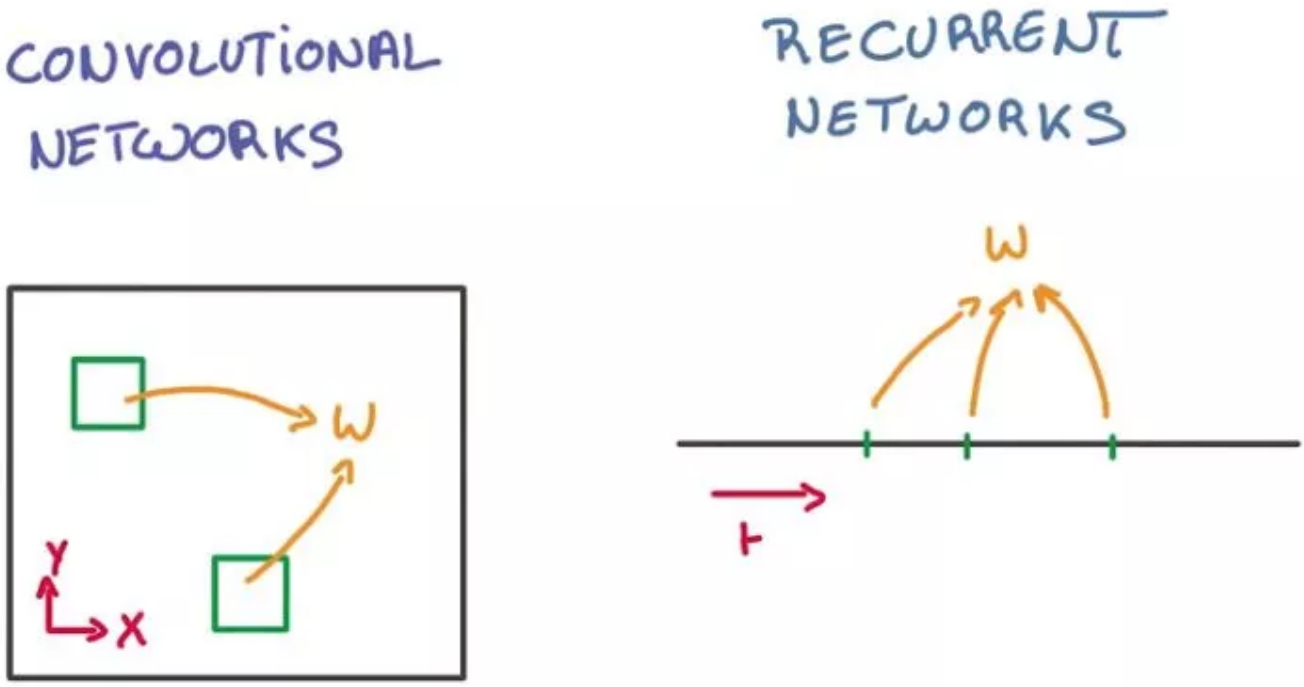
例2：演奏音乐时，乐器将力转成相应的震动产生声音，而整个演奏拥有一个主旋律贯穿全曲。其中乐器的物理特性就相当于参数，同一乐器在各个时刻物理特性在各个时刻都是共享的。其内在也有一个隐藏的主旋律基准（主题），旋律信息（上一个状态乘与主题）与音乐信息（输入称与参数）共同决定下一时刻的实际声音。

而捏陶瓷的例子可能更容易体会共享特性对于数据量的影响，不同角度相当于不同的时刻

- 若用前馈网络：网络训练过程相当于不用转盘，而是徒手将各个角度捏成想要的形状。不仅工作量大，效果也难以保证。
- 若用递归网络：网络训练过程相当于在不断旋转的转盘上，以一种手势捏造所有角度。工作量降低，效果也可保证。

CNN 和RNN的共享参数:

我们需要记住的是，深度学习是怎么减少参数的，很大原因就是参数共享，而**CNN是 在空间上共享参数，RNN是在时间上（顺序上）共享参数**:



这个图虽然丑，但是表明了RNN和CNN在不同角度上的参数共享

rnn的训练

对于RNN的训练和对传统的ANN训练一样。同样使用BP误差反向传播算法，不过有一点区别。如果将RNN进行网络展开，那么参数W,U,V是共享的，而传统神经网络却不是

的。并且在使用梯度下降算法中，每一步的输出不仅依赖当前步的网络，并且还用前面若干步网络的状态。比如，在t=4时，我们还需要向后传递三步，以及后面的三步都需要加上各种的梯度。该学习算法称为Backpropagation Through Time (BPTT)。需要注意的是，在普通RNN训练中，BPTT无法解决长时依赖问题(即当前的输出与前面很长的一段序列有关，一般超过十步就无能为力了)，因为BPTT会带来所谓的梯度消失或梯度爆炸问题(the vanishing/exploding gradient problem)。当然，有很多方法去解决这个问题，如LSTM便是专门应对这种问题的。

> 因为BPTT具体推导的话占篇幅较大，之后在看看单独一篇来写吧！这里先做个简介。

RNN几种架构

然后来说说几种用RNN组成的常用架构，如下图：

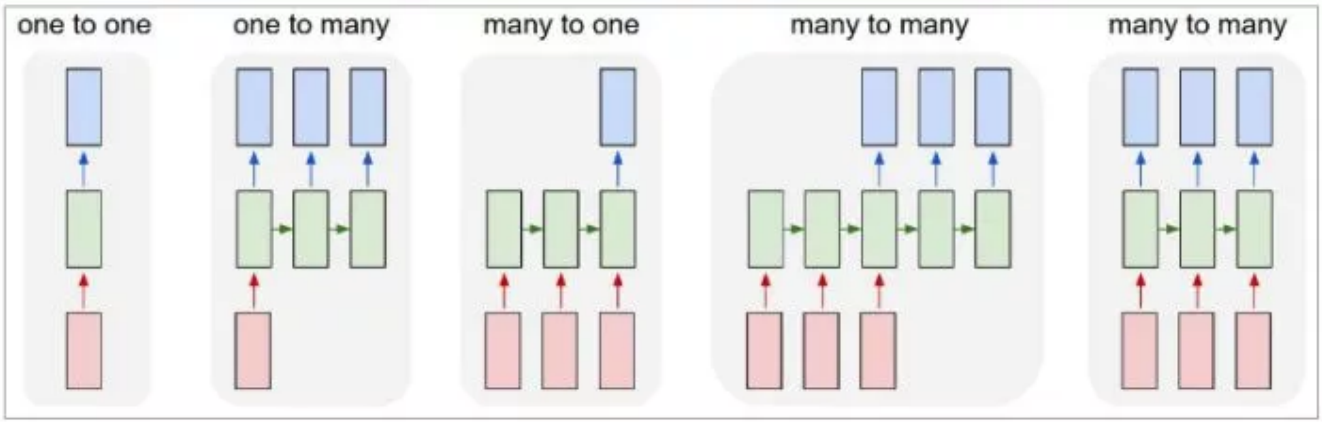
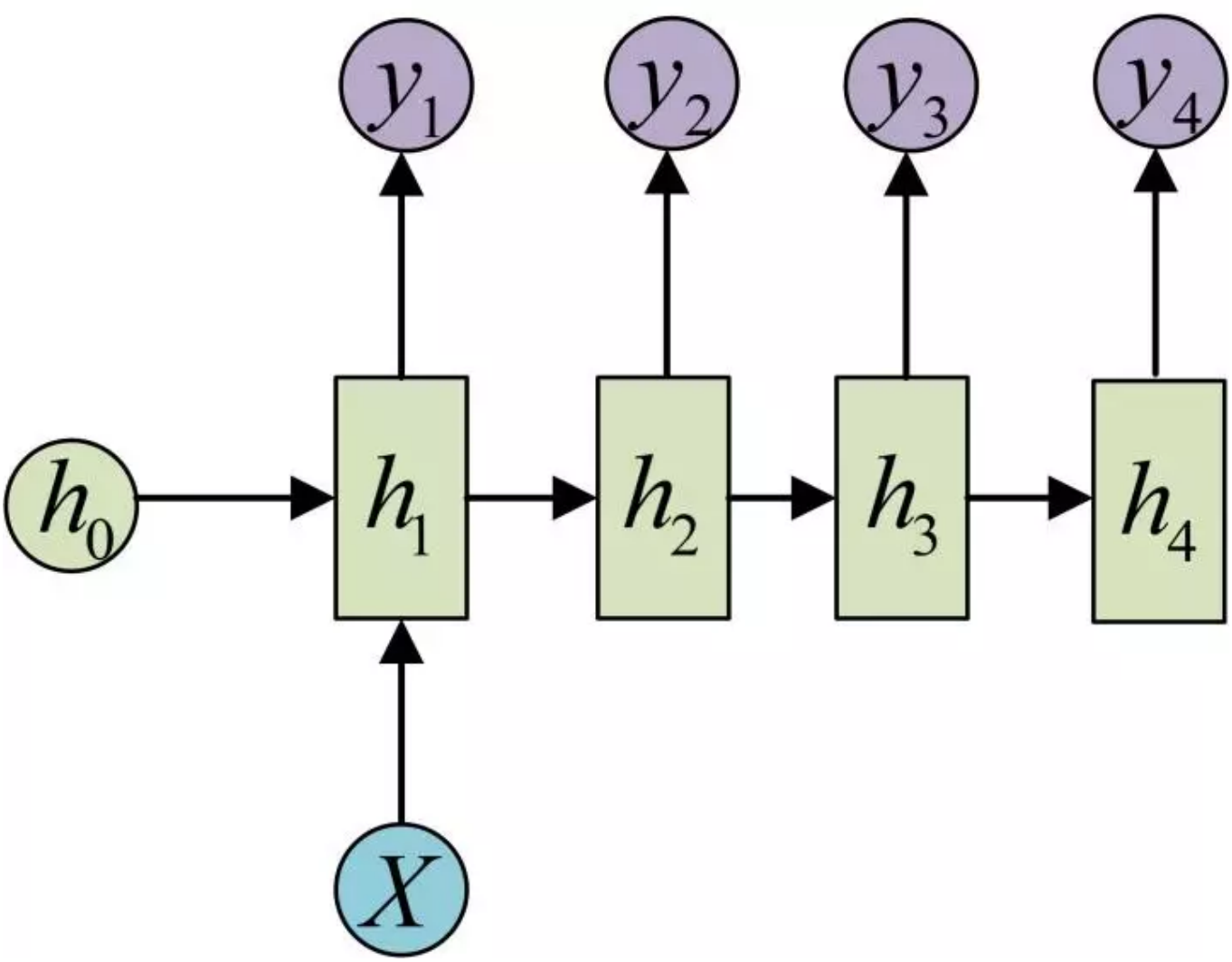


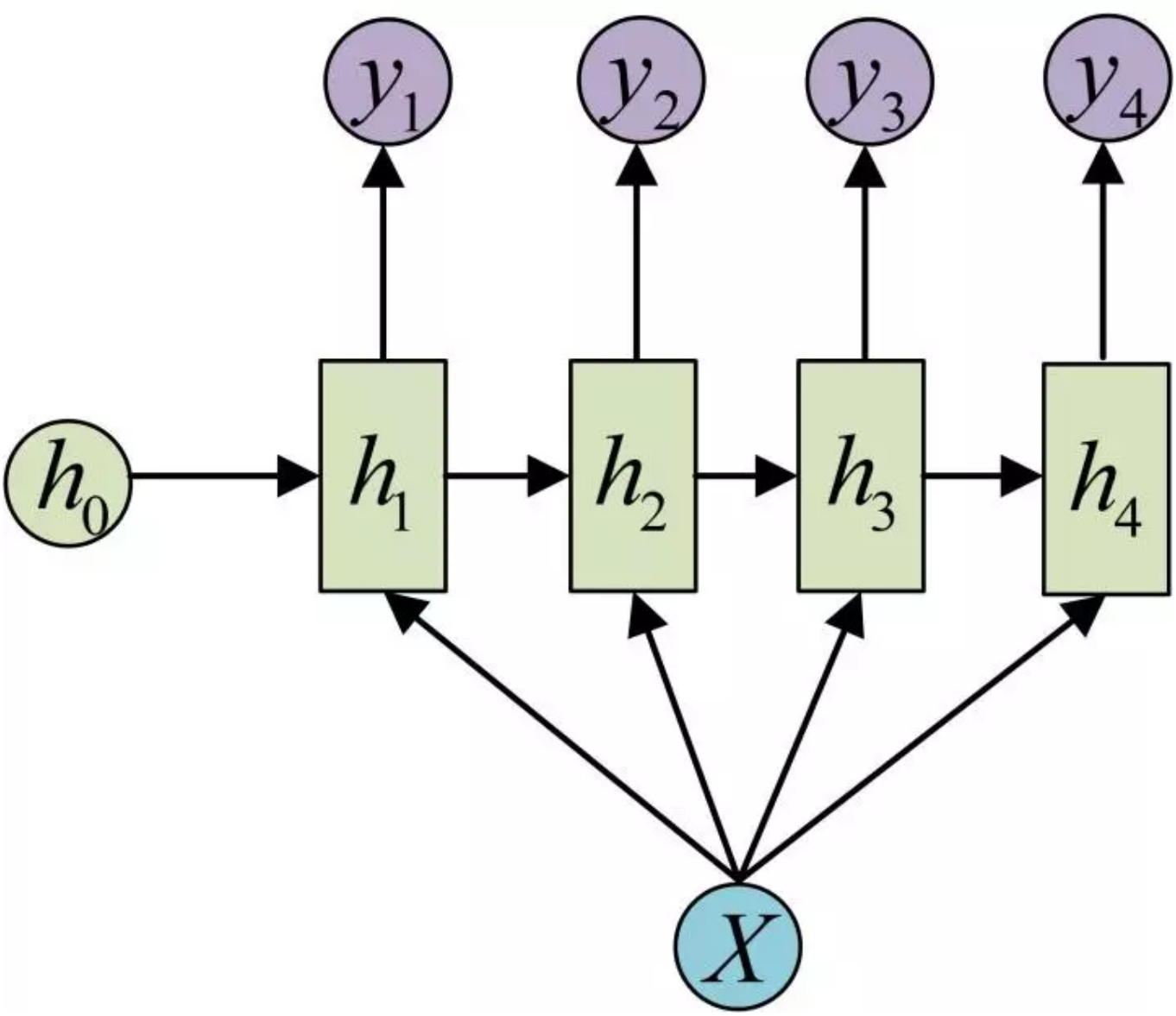
图1是普通的单个神经网络，图2是把单一输入转化为序列输出。 图3是把序列输入转化为单个输出 图4是把序列转化为序列，也就是seq2seq的做法。 图5是无时差的序列到序列转化，可以作为普通得语言模型，下面再说说几个比较重要的架构：

1 to N：

这种情况有两种方式，一种是只在序列开始进行输入计算



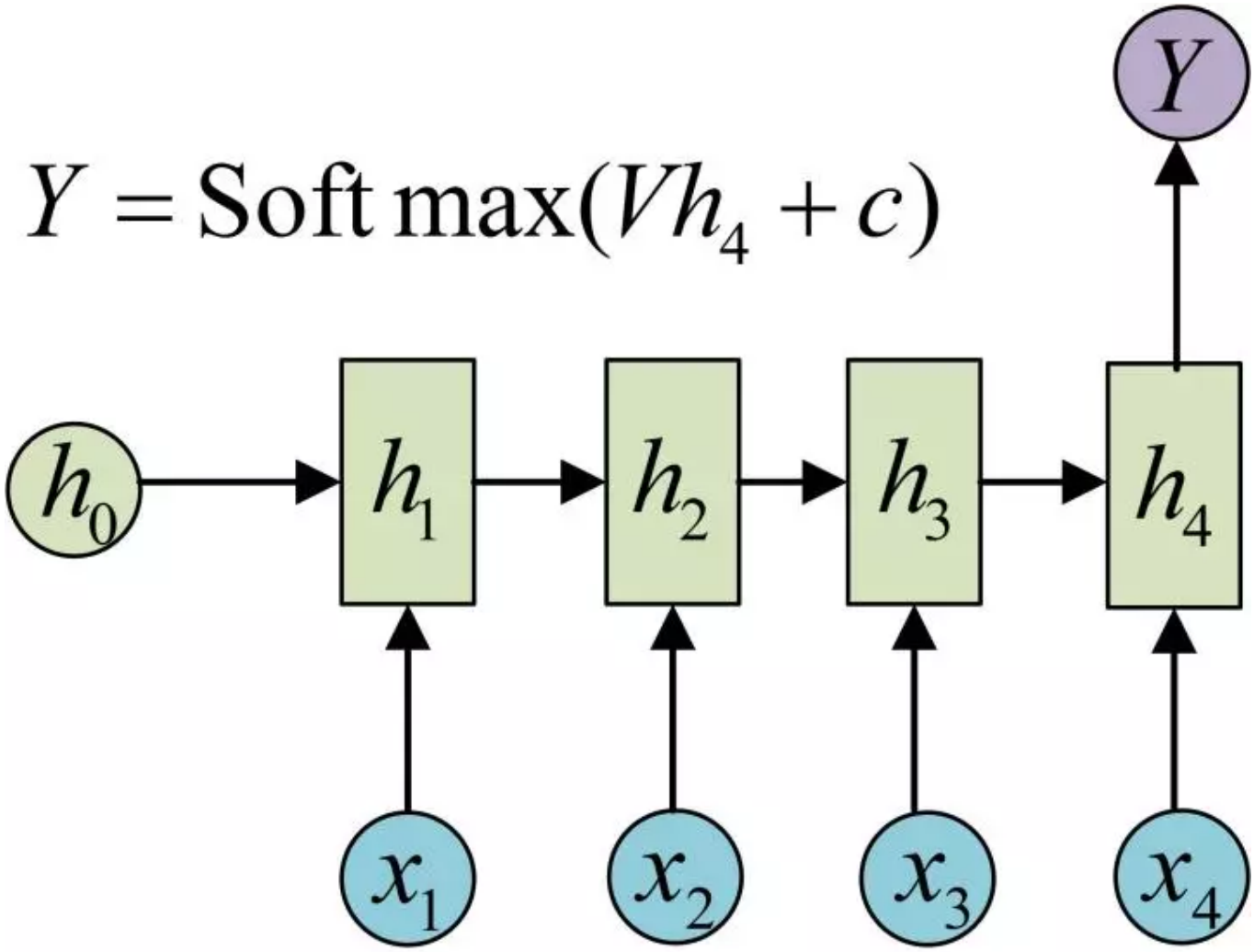
还有一种结构是把输入信息X作为每个阶段的输入：



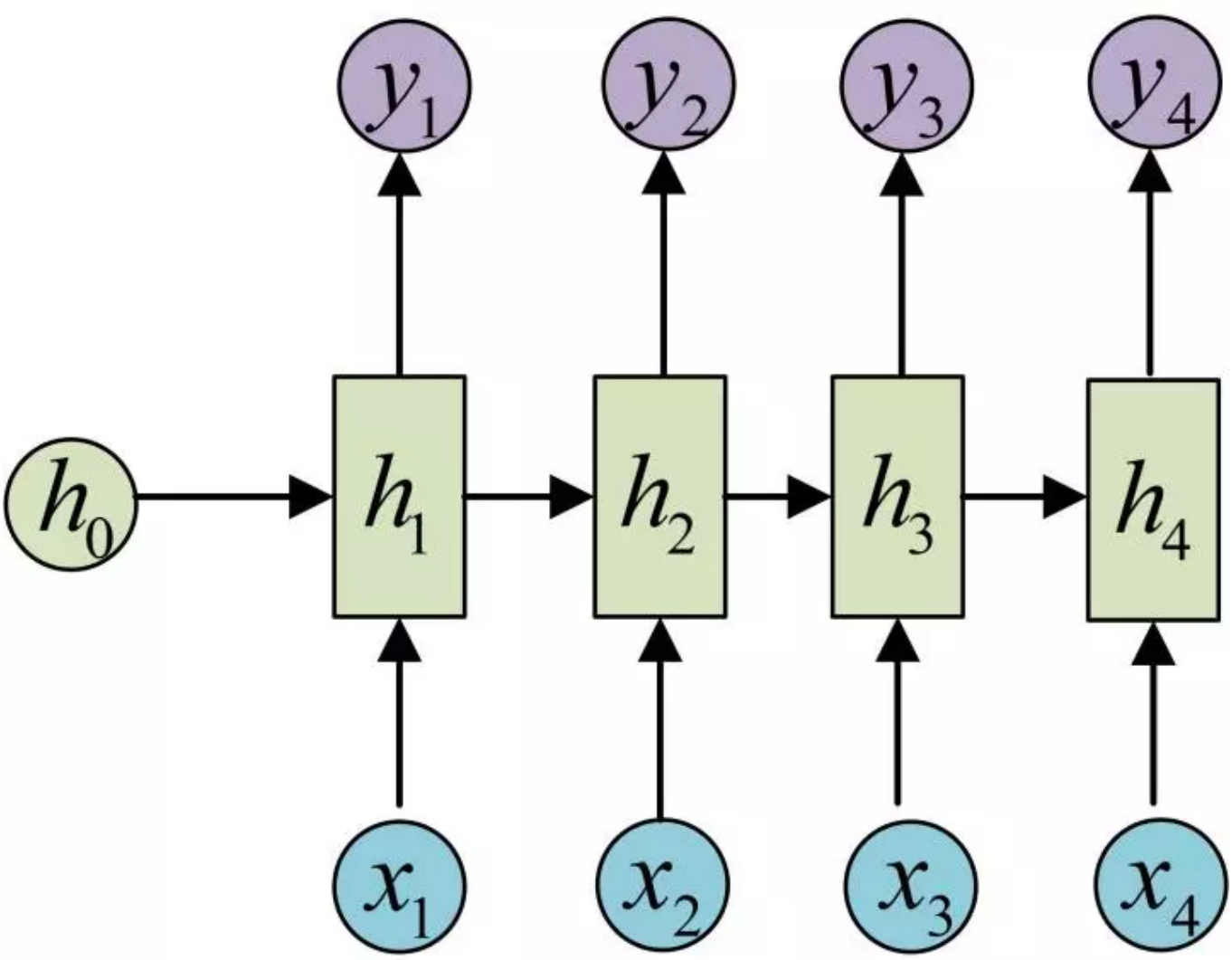
这种1toN的结构可以处理的问题有挺多的，比如图片标注，输出的X是图像的特征，而输出的y序列是一段句子或者从类别生成语音或音乐，输入的x是类别，输出的y是一串文字。

N to 1

输入是一个序列，输出是一个单独的值而不是序列。这种结构通常用来处理序列分类问题。如输入一段文字判别它所属的类别，输入一个句子判断其情感倾向，输入一段文档并判断它的类别等等。具体如下图：



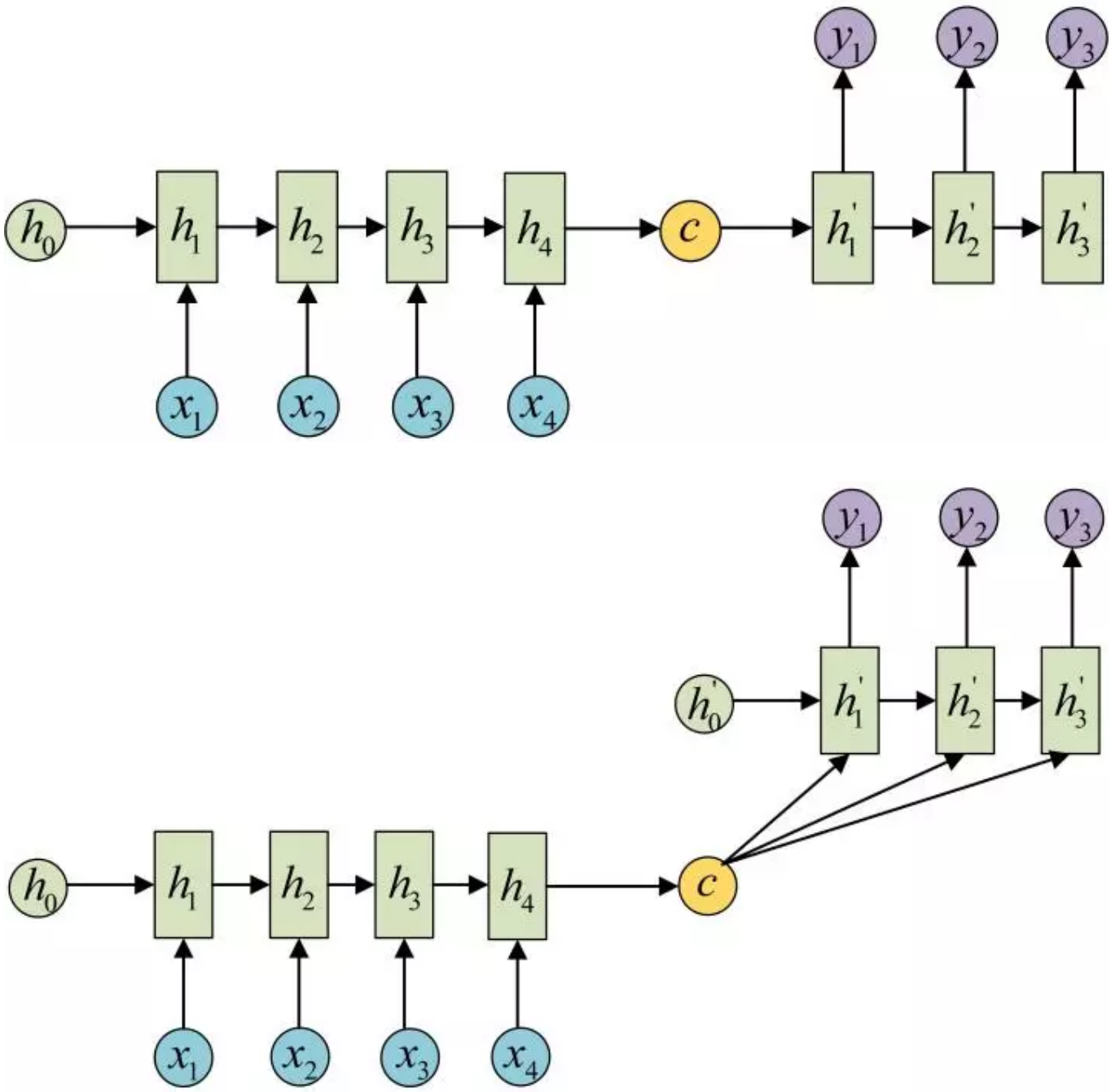
N to N



输入和输出序列是等长的。这种可以作为简单的Char RNN可以用来生成文章，诗歌，甚至是代码，非常有意思）。

N to M

这种结构又叫Encoder-Decoder模型，也可以称之为Seq2Seq模型。在实现问题中，我们遇到的大部分序列都是不等长的，如机器翻译中，源语言和目标语言的句子往往并没有相同的长度。而Encoder-Decoder结构先将输入数据编码成一个上下文向量c，之后在通过这个上下文向量输出预测序列。

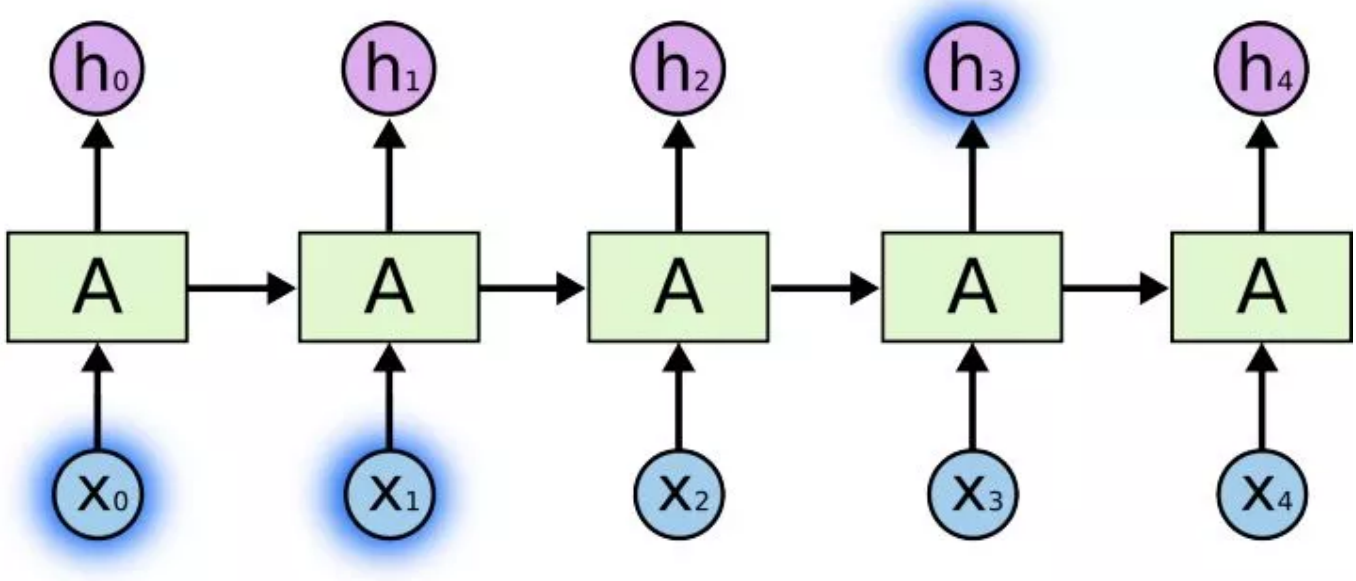


> 注意，很多时候只用上下文向量C效果并不是很好，而attention技术很大程度弥补了这点。

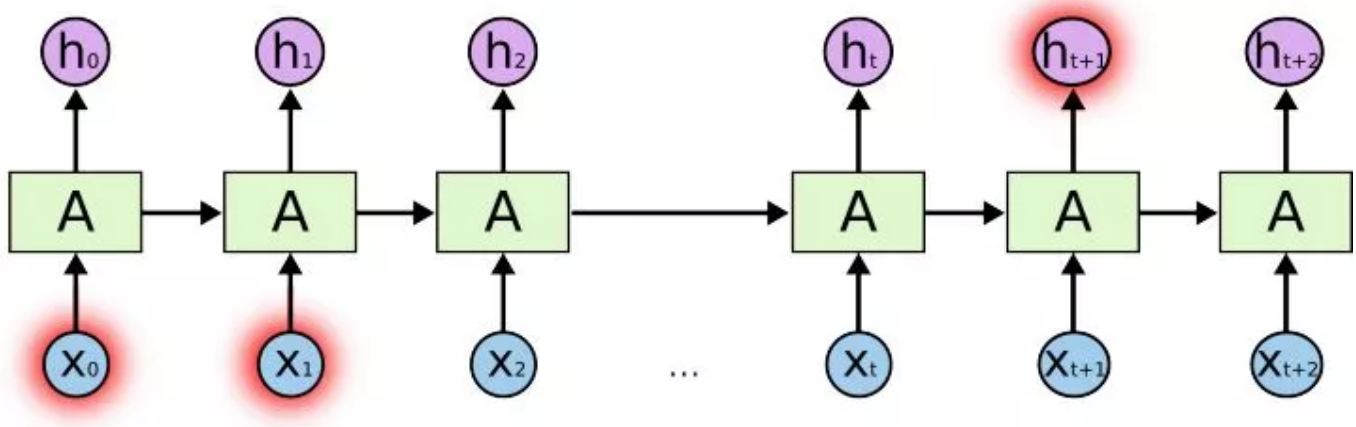
seq2seq的应用的范围非常广泛，机器翻译，文本摘要，阅读理解，对话生成....，之后会找机会详细说明其中的知识，包括各种attention，不同结构等。

Long-Term依赖问题

有时候，我们只需要查看最近的信息来执行现在的任务，例如，考虑一个语言模型试图基于先前的词预测下一个词。如果我们要预测“the clouds are in the sky”，我们不需要其他更遥远的上下文 —— 非常明显，下一个词就应该是sky。在这样的例子中，相关信息和目的地之间的距离是很小的。RNN可以学着去使用过去的信息。



但也有一些情况是我们需要更多上下文的。考虑预测这个句子中最后一个词：“I grew up in France… I speak fluent French.” 最近的信息表明下一个词可能是一种语言的名字，但如果我们想要找出是哪种语言，我们需要从更久远的地方获取France的上下文。相关信息和目标之间的距离完全可能是非常巨大的。而不幸的是，随着距离的增大，RNN变得不能够连接信息。



理论上，RNN是绝对能够处理这样的“长期依赖的”。人类可以仔细地从这些词中找到参数然后解决这种形式的一些雏形问题。然而，实践中，RNN似乎不能够学习到这些。Hochreiter (1991) [German] 和 Bengio, et al. 1994年曾探索过这个问题，他们发现了一些非常根本的导致RNN难以生效的原因。幸运的是，LSTM或GRU没有这个问题！也就是说不会产生由于太多层导致的梯度爆炸或者梯度消亡的问题！在之后文章中，将为大家讲述lstm和GRU，敬请期待！