Microsoft

Reactor

一起学人工智能系列
- 循环神经网络

2021-12-22

Map


Stockholm
COMING SOON
London
Redmond
Toronto
New York
San Francisco
Tel Aviv
Shanghai
Abu Dhabi
Bangalore
Sao Paulo
Sydney

个人介绍

Kinfey Lo – (卢建晖)

Microsoft Cloud Advocate

前微软MVP、Xamarin MVP和微软RD，拥有超过10年的云原生、人工智能和移动应用经验，为教育、金融和医疗提供应用解决方案。 Microsoft Iginte，Teched 会议讲师，Microsoft AI 黑客马拉松教练，目前在微软，为技术人员和不同行业宣讲技术和相关应用场景。

爱编程(Python， C#，TypeScript，Swift，Rust，Go ）

专注于人工智能，云原生，跨平台移动开发

Github : https://github.com/kinfey

Email : kinfeylo@microsoft.com **Blog :** https://blog.csdn.net/kinfey

Twitter : @Ljh8304

卷积神经网络原理

引言

循环神经网络实际上前馈全连接神经网络的一种扩展，如果你已经掌握了全连接神经网络中的算法、公式推导等知识，那么学习循环神经网络会很容易。

如果说全连接网络是学习静态数据的非线性特征的，那么循环神经网络就是学习动态序列数据的非线性特征的。

所有的神经网络的输入都是一个或者一批静态的数据，比如一个人的身高、体重、年龄、性别等组成的特征值用于表示一个人当前的属性，这些属性是采样时获得的，并且会保持相对稳定，可以用这些属性通过前馈神经网络来预测一个人的健康状况。再次输入的下一个数据会是另外一个人的特征值，与前一个人丝毫不相关。

或者输入的是一张青蛙的图片，通过卷积神经网络来判断图片中的物体的类别。而下一张图片可能会是另外一只青蛙的图片，或者干脆变成了一张汽车的图片。

而在自然界中，还有很多随着时间而变化的数据需要处理，比如，对一个人来说，在不同的年龄会有不同的身高、体重、健康状况，只有性别是固定的。如果需要根据年龄来预测此人的健康状况，则需要每年对此人的情况进行一次采样，按时间排序后记录到数据库中。

另外一个例子是如果想从一只青蛙的跳跃动作中分析出其跳跃的高度和距离，则需要获得一段视频，然后从视频的每一帧图片中获得青蛙的当前位置和动作。

循环神经网络的发展简史

循环神经网络（RNN，Recurrent Neural Network）的历史可以简单概括如下：

1933年，西班牙神经生物学家Rafael Lorente de Nó发现大脑皮层（cerebral cortex）的解剖结构允许刺激在神经回路中循环传递，并由此提出反响回路假设（reverberating circuit hypothesis）。

1982年，美国学者John Hopfield使用二元节点建立了具有结合存储（content-addressable memory）能力的神经网络，即Hopfield神经网络。

1986年，Michael I. Jordan基于Hopfield网络的结合存储概念，在分布式并行处理（parallel distributed processing）理论下建立了新的循环神经网络，即Jordan网络。

1990年，Jeffrey Elman提出了第一个全连接的循环神经网络，Elman网络。Jordan网络和Elman网络是最早出现的面向序列数据的循环神经网络，由于二者都从单层前馈神经网络出发构建递归连接，因此也被称为简单循环网络（Simple Recurrent Network, SRN）。

1990年，Paul Werbos提出了循环神经网络的随时间反向传播（BP Through Time，BPTT），BPTT被沿用至今，是循环神经网络进行学习的主要方法。

1991年，Sepp Hochreiter发现了循环神经网络的长期依赖问题（long-term dependencies problem），大量优化理论得到引入并衍生出许多改进算法，包括神经历史压缩器（Neural History Compressor, NHC）、长短期记忆网络（Long Short-Term Memory networks, LSTM）、门控循环单元网络（Gated Recurrent Unit networks, GRU）、回声状态网络（echo state network）、独立循环神经网络（Independent RNN）等。

循环神经网络的发展简史

循环神经网络（RNN，Recurrent Neural Network）的历史可以简单概括如下：

1933年，西班牙神经生物学家Rafael Lorente de Nó发现大脑皮层（cerebral cortex）的解剖结构允许刺激在神经回路中循环传递，并由此提出反响回路假设（reverberating circuit hypothesis）。

1982年，美国学者John Hopfield使用二元节点建立了具有结合存储（content-addressable memory）能力的神经网络，即Hopfield神经网络。

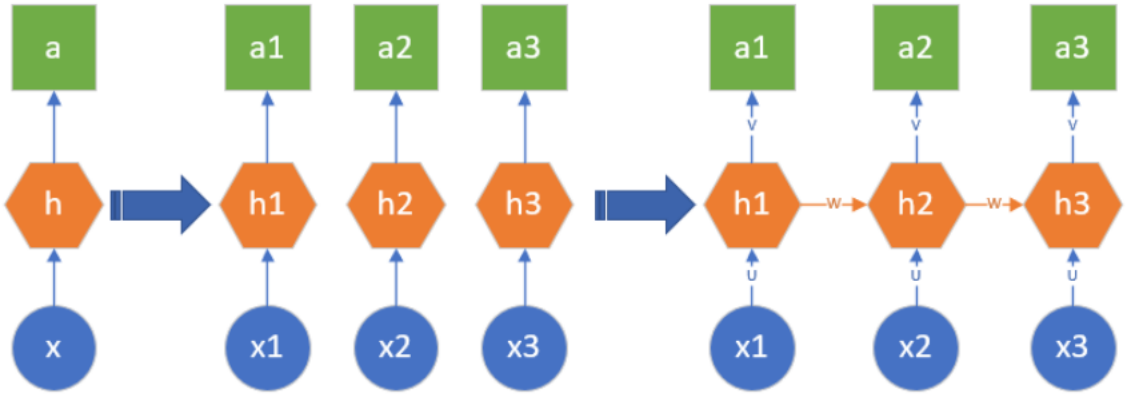
1986年，Michael I. Jordan基于Hopfield网络的结合存储概念，在分布式并行处理（parallel distributed processing）理论下建立了新的循环神经网络，即Jordan网络。

1990年，Jeffrey Elman提出了第一个全连接的循环神经网络，Elman网络。Jordan网络和Elman网络是最早出现的面向序列数据的循环神经网络，由于二者都从单层前馈神经网络出发构建递归连接，因此也被称为简单循环网络（Simple Recurrent Network, SRN）。

1990年，Paul Werbos提出了循环神经网络的随时间反向传播（BP Through Time，BPTT），BPTT被沿用至今，是循环神经网络进行学习的主要方法。

1991年，Sepp Hochreiter发现了循环神经网络的长期依赖问题（long-term dependencies problem），大量优化理论得到引入并衍生出许多改进算法，包括神经历史压缩器（Neural History Compressor, NHC）、长短期记忆网络（Long Short-Term Memory networks, LSTM）、门控循环单元网络（Gated Recurrent Unit networks, GRU）、回声状态网络（echo state network）、独立循环神经网络（Independent RNN）等。

从前馈神经网络到循环神经网络的演化过程

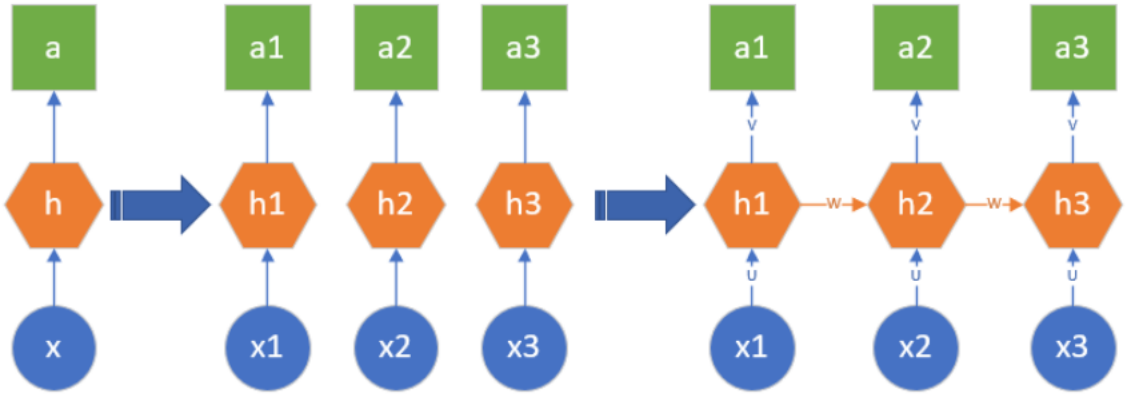


最左侧的是前馈神经网络的概括图，即，根据一个静态的输入数据x，经过隐层h的计算，最终得到结果a。这里的h是全连接神经网络或者卷积神经网络，a是回归或分类的结果。

当遇到序列数据的问题后（假设时间步数为3），可以建立三个前馈神经网络来分别处理t=1、t=2、t=3的数据，即x1、x2、x3

但是两个时间步之间是有联系的，于是在隐层h1、h2、h3之间建立了一条连接线，实际上是一个矩阵W
根据序列数据的特性，可以扩充时间步的数量，在每个相邻的时间步之间都会有联系

从前馈神经网络到循环神经网络的演化过程



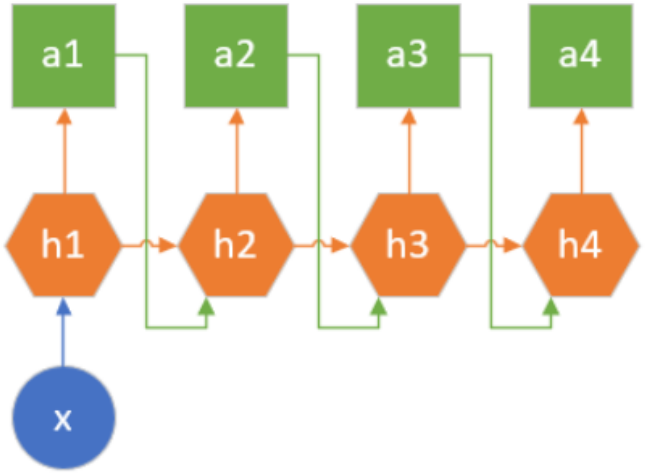
如果仅此而已的话，还不能称之为循环神经网络，只能说是多个前馈神经网络的堆叠而已。在循环神经网络中，只有三个参数：

U：是x到隐层h的权重矩阵
V：是隐层h到输出a的权重矩阵
W：是相邻隐层h之间的权重矩阵

请注意这三个参数在不同的时间步是共享的，三个U其实是同一个矩阵，三个V是同一个矩阵，两个W是同一个矩阵。这样的话，无论有多少个时间步，都可以像折扇一样“折叠”起来，用一个“循环”来计算各个时间步的输出，这才是“循环神经网络”的真正含义。

循环神经网络的结构和典型用途

一对多的结构

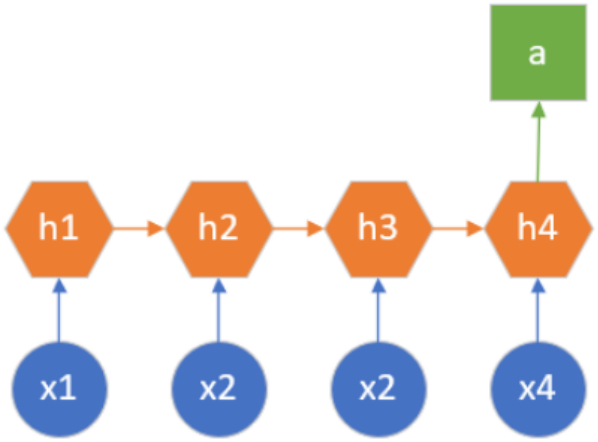


在国外，用户可以指定一个风格，或者一段旋律，让机器自动生成一段具有巴赫风格的乐曲。在中国，有藏头诗的娱乐形式，比如以“春”字开头的一句五言绝句可以是“春眠不觉晓”、“春草细还生”等等。这两个例子都是只给出一个输入，生成多个输出的情况

这种情况的特殊性在于，第一个时间步生成的结果要做为第二个时间步的输入，使得前后有连贯性。图中只画出了4个时间步，在实际的应用中，如果是五言绝句，则有5个时间步；如果是音乐，则要指定小节数，比如40个小节，则时间步为40。

循环神经网络的结构和典型用途

多对一的结构



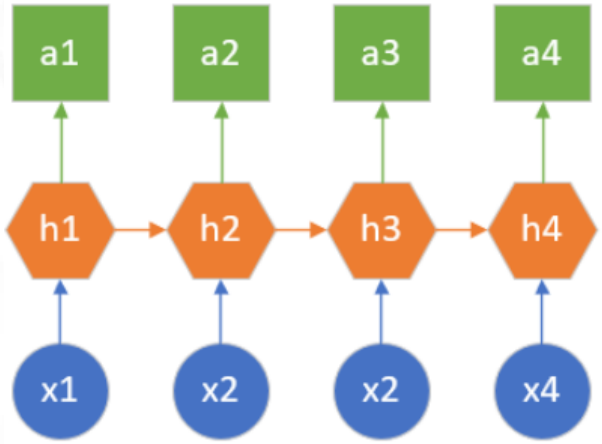
在阅读一段影评后，会判断出该观众对所评价的电影的基本印象如何，比如是积极的评价还是消极的评价，反映在数值上就是给5颗星还是只给1颗星。在这个例子中，输入是一段话，可以拆成很多句或者很多词组，输出则是一个分类结果。这是一个多个输入单个输出的形式

图中x可以看作很多连续的词组，依次输入到网络中，只在最后一个时间步才有一个统一的输出。另外一种典型的应用就是视频动作识别，输入连续的视频帧（图片形式），输出是分类结果，比如“跑步”、“骑车”等等动作。

还有一个很吸引人的应用就是股票价格的预测，输入是前10天的股票基本数据，如每天的开盘价、收盘价、交易量等，而输出是明天的股票的收盘价，这也是典型的多对一的应用。但是由于很多其它因素的干扰，股票价格预测具有很大的不确定性。

循环神经网络的结构和典型用途

多对多(输入输出等量)



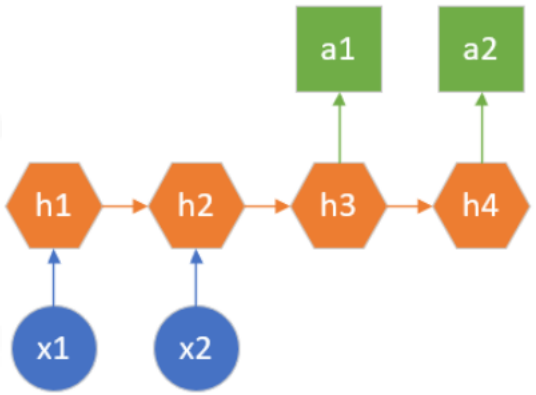
这种结构要求输入的数据时间步的数量和输出的数据的时间步的数量相同

比如想分析视频中每一帧的分类，则输入100帧输入，输出是对应的100个分类结果。另外一个典型应用就是基于字符的语言模型，比如对于英文单词“hello”来说，当第一个字母是h时，计算第二个字母是e的概率，以此类推，则输入是“hell”四个字母，输出是“ello”四个字母的概率。

在中文中，对联的生成问题也是使用了这种结构，如果上联是“风吹水面层层浪”七个字，则下联也一定是七个字，如“雨打沙滩点点坑”。

循环神经网络的结构和典型用途

多对多(输入输出不等量)



这是循环神经网络最重要的一个变种，又叫做编码解码（Encoder-Decoder）模型，或者序列到序列（seqence to seqence）模型

以机器翻译任务举例，源语言和目标语言的句子通常不会是相同的长度，为此，此种结构会先把输入数据编码成一个上下文向量，在h2后生成，做为h3的输入。此时，h1和h2可以看做是一个编码网络，h3和h4看做是一个解码网络。解码网络拿到编码网络的输出后，进行解码，得到目标语言的句子。

由于这种结构不限制输入和输出的序列长度，所以应用范围广泛，类似的应用还有：

文本摘要：输入是一段文本，输出是摘要，摘要的字数要比正文少很多。

阅读理解：输入是文章和问题，输出是问题答案，答案一般都很简短。

语音识别，输入是语音信号序列，输出是文字序列，输入的语音信号按时间计算长度，而输出按字数计算长度，根本不是一个量纲。

两个时间步的循环神经网络

案例引入

我们先用一个最简单的序列问题来了解一下循环神经网络的基本运作方式。

假设有一个随机信号发射器，每秒产生一个随机信号，随机值为(0,1)之间。信号发出后，碰到一面墙壁反射回来，来回的时间相加正好是1秒，于是接收器就收到了1秒钟之前的信号。对于接收端来说，接收到的数据序列如表19-1。

表19-1 每一时刻的发射信号和回波信号

时刻	t1	t2	t3	t4	t5	t6	...
发射随机信号X	0.35	0.46	0.12	0.69	0.24	0.94	...
接收回波信号Y	0	0.35	0.46	0.12	0.69	0.24	...

具体地描述此问题：当接收端接收到两个连续的值，如0.35、0.46时，系统响应为0.35；下一个时间点接收到了0.12，考虑到上一个时间点的0.46，则二者组合成0.46、0.12序列，此时系统响应为0.46；依此类推，即接收到第二个数值时，总要返回相邻的第一个数值。

我们可以把发射信号看作X，把接收信号看作是Y，则此问题变成了给定样本X和标签值Y，训练一个神经网络，令其当接收到两个序列的值时，总返回第一个值。

四个时间步的循环神经网络

案例引入

在加减法运算中，总会遇到进位或者退位的问题，我们以二进制为例，比如13-6=7这个十进制的减法，变成二进制后如下所示：

```
13 - 6 = 7
====================
  x1: [1, 1, 0, 1]
- x2: [0, 1, 1, 0]
------------------
  y:  [0, 1, 1, 1]
====================
```

- 被减数13变成了[1, 1, 0, 1]
- 减数6变成了[0, 1, 1, 0]
- 结果7变成了[0, 1, 1, 1]

在减法过程中：

- x1和x2的最后一位是1和0，相减为1
- 倒数第二位是0和1，需要从前面借一位，相减后得1
- 倒数第三位本来是1和1，借位后变成了0和1，再从前面借一位，相减后得1
- 倒数第四位本来是1和0，借位后是0和0，相减为0

也就是说，在减法过程中，后面的计算会影响前面的值，所以必须逐位计算，这也就是时间步的概念，所以应该可以用循环神经网络的技术来解决。

通用的循环神经网络

不同场景下的循环神经网络参数

	回波检测问题	二进制减法问题	PM2.5预测问题
时间步	2	4	用户指定参数
网络输出类别	回归	二分类	多分类
分类函数	无	Logistic函数	Softmax函数
损失函数	均方差	二分类交叉熵	多分类交叉熵
时间步输出	最后一个	每一个	最后一个
批大小	1	1	用户指定参数
有无偏移值	有	无	有

1.既可以支持分类网络（二分类和多分类），也可以支持回归网络；

2.每一个时间步可以有输出并且有监督学习信号，也可以只在最后一个时间步有输出；

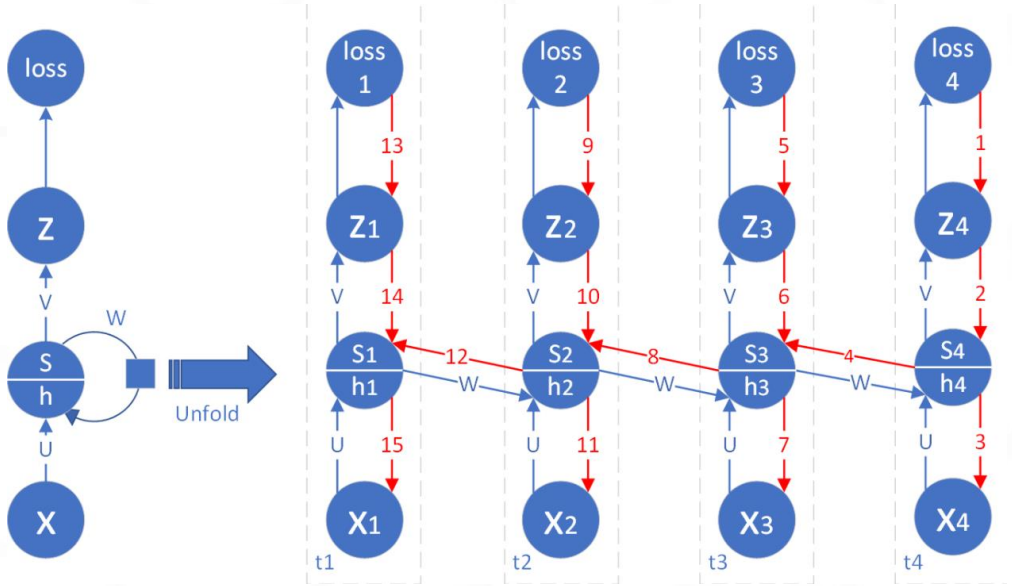
3.第一个时间步的前向计算中不包含前一个时间步的隐层状态值（因为前面没有时间步）；

4.最后一个时间步的反向传播中不包含下一个时间步的回传误差（因为后面没有时间步）；

5.可以指定超参数进行网络训练，如：学习率、批大小、最大训练次数、输入层尺寸、隐层神经元数量、输出层尺寸等等；

6.可以保存训练结果并可以在以后加载参数，避免重新训练。

全输出网络通过时间反向传播



正向计算过程为：

$$h_t = x_t \cdot U + s_{t-1} \cdot W + b_u$$

只有在时间步t1时，公式1中的s_{t-1}为0。

后续过程为：

$$s_t = \sigma(h_t)$$

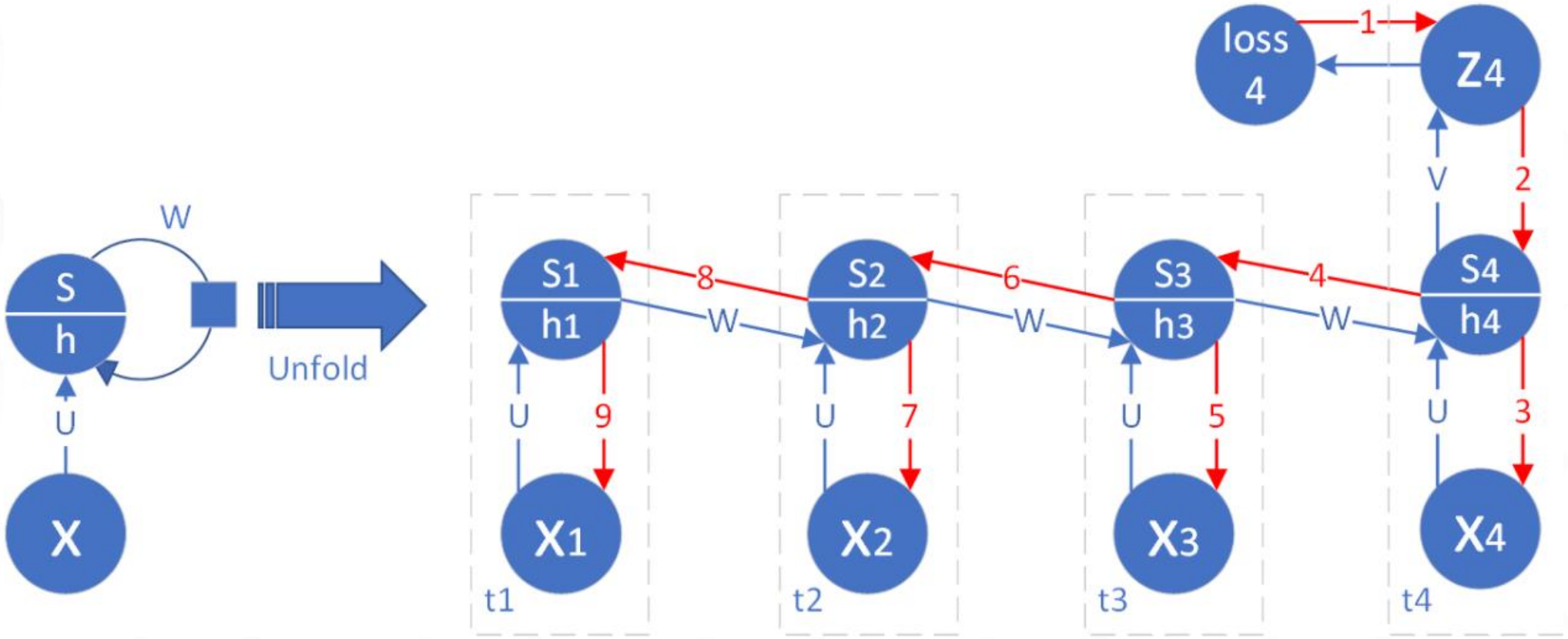
$$z_t = V \cdot s_t + b_v$$

$$a_t = C(z_t)$$

$$loss_t = L(a_t, y_t)$$

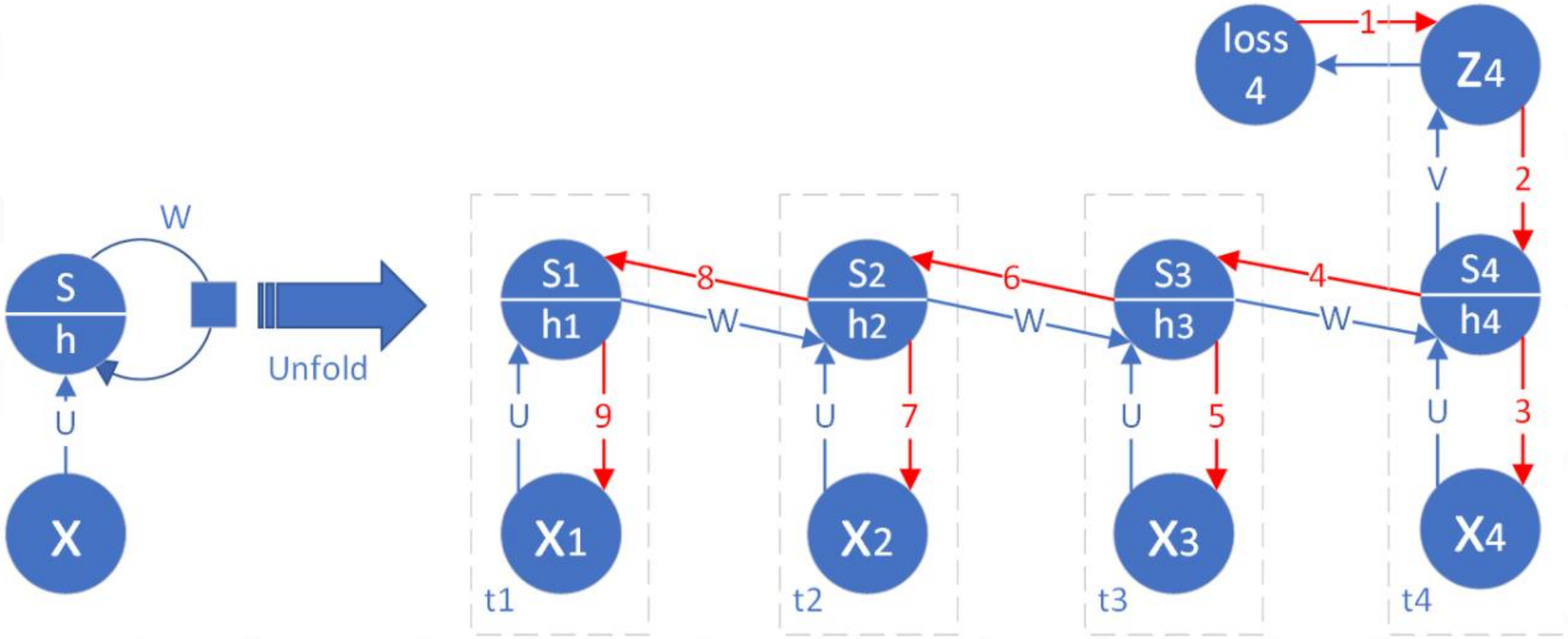
时间的反向传播（BPTT，Back Propagation Through Time）

单输出网络通过时间的反向传播

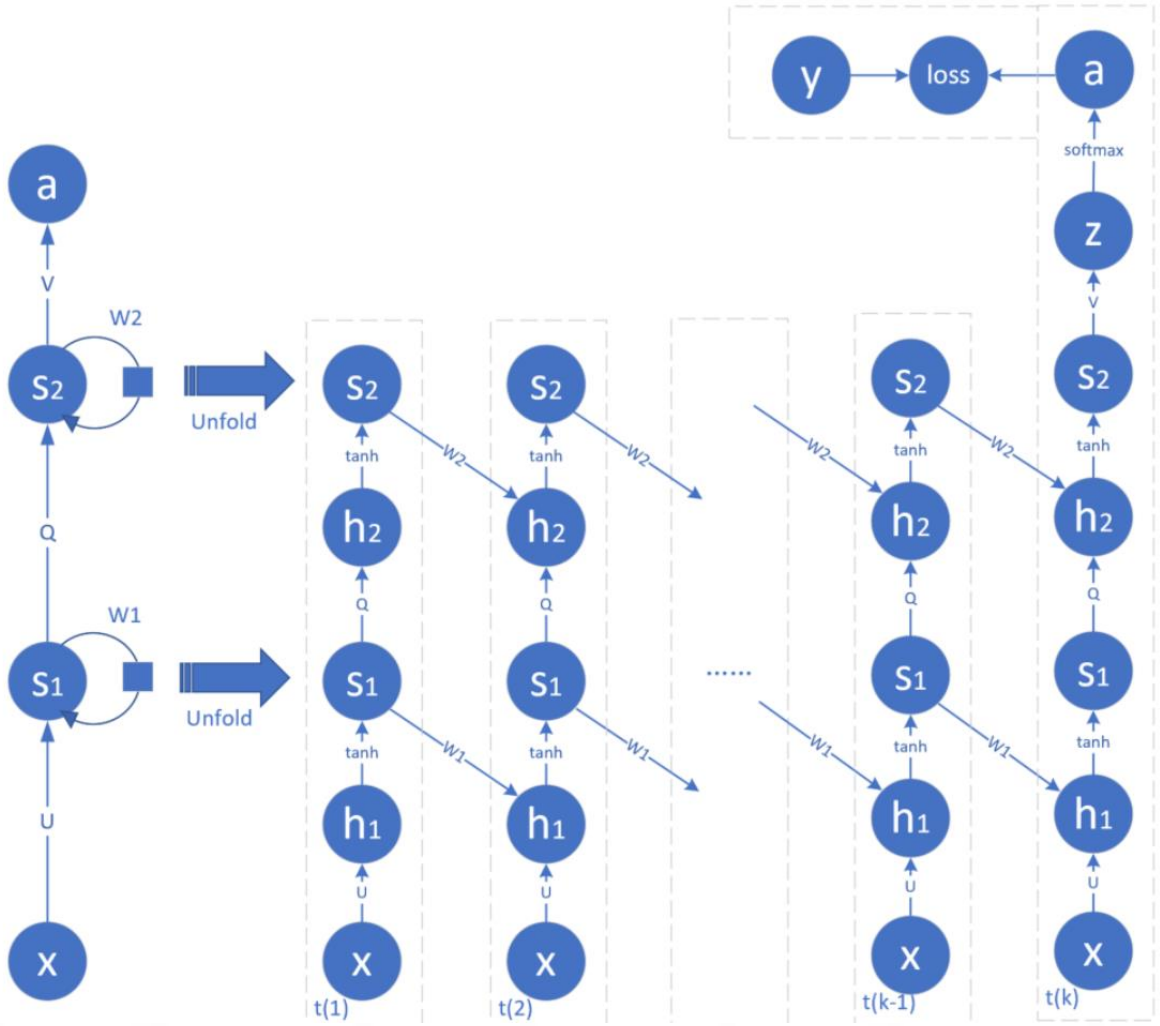


深度循环神经网络

单输出网络通过时间的反向传播



深度循环神经网络的结构图



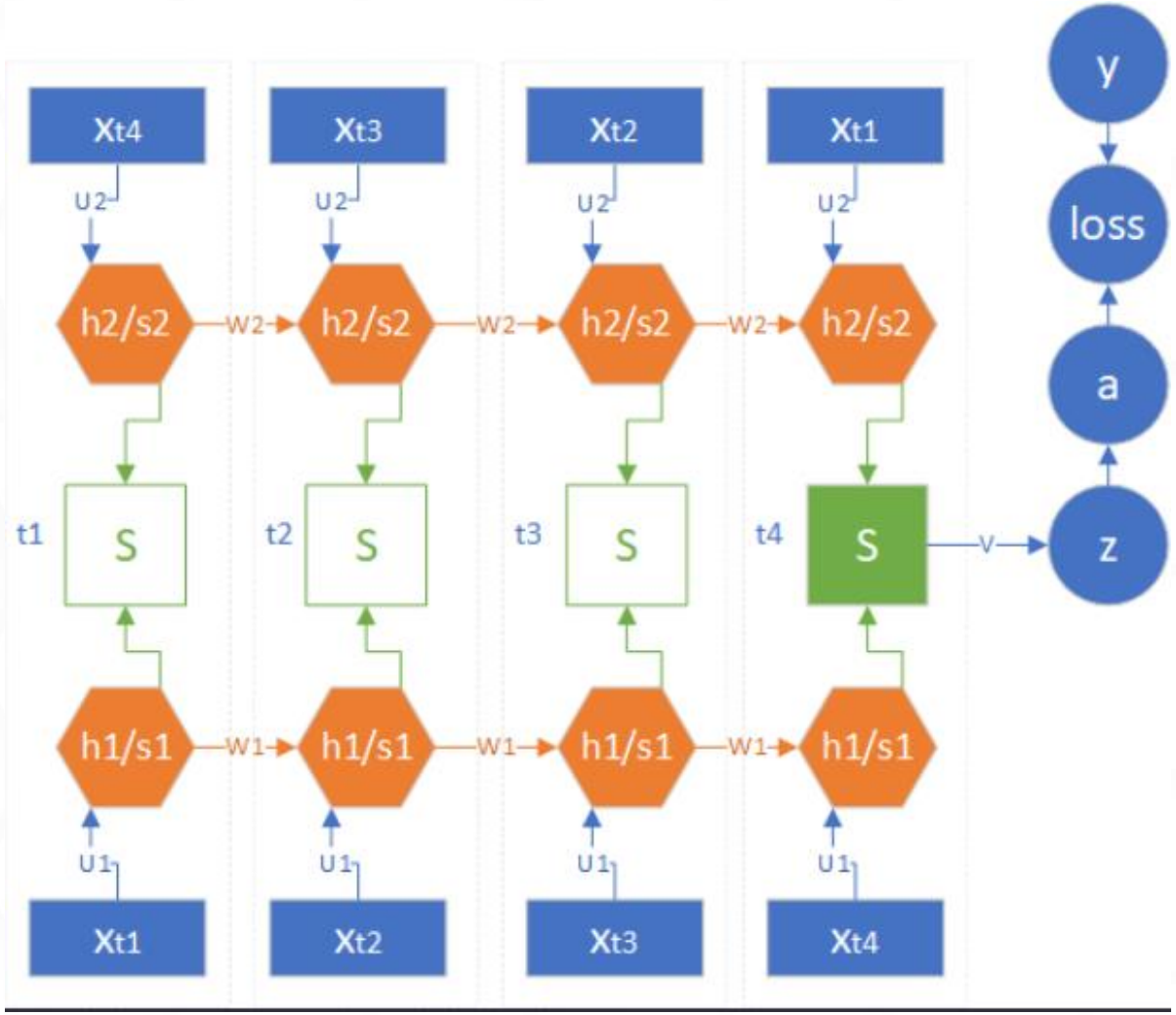
双向循环神经网络

深度循环神经网络的结构图

在一个语音识别的模型中，可能前面的一个词听上去比较模糊，会产生多个猜测，但是后面的词都很清晰，于是可以用后面的词来为前面的词提供一个最有把握（概率最大）的猜测。再比如，在手写识别应用中，前面的笔划与后面的笔划是相互影响的，特别是后面的笔划对整个字的识别有较大的影响。

- 前向计算：是指神经网络中通常所说的前向计算，包括正向循环的前向计算和逆向循环的前向计算。
- 反向传播：是指神经网络中通常所说的反向传播，包括正向循环的反向传播和逆向循环的反向传播。
- 正向循环：是指双向循环神经网络中的从左到右时间步。在正向过程中，会存在前向计算和反向传播。
- 逆向循环：是指双向循环神经网络中的从右到左时间步。在逆向过程中，也会存在前向计算和反向传播。

双向循环神经网络图



高级循环神经网络概述

传统循环神经网络的不足

循环神经网络弥补了前馈神经网络的不足，可以更好的处理时序相关的问题，扩大了神经网络解决问题的范围

但传统的循环神经网络也有自身的缺陷，由于容易产生梯度爆炸和梯度消失的问题，导致很难处理长距离的依赖。传统神经网络模型，不论是一对多、多对一、多对多，都很难处理不确定序列输出的问题，一般需要输出序列为1，或与输入相同。在机器翻译等问题上产生了局限性。

长短时记忆网络（LSTM）
长短时记忆网络（LSTM）是最先提出的改进算法，由于门控单元的引入，从根本上解决了梯度爆炸和消失的问题，使网络可以处理长距离依赖。

门控循环单元网络（GRU）
LSTM网络结构中有三个门控单元和两个状态，参数较多，实现复杂。为此，针对LSTM提出了许多变体，其中门控循环单元网络是最流行的一种，它将三个门减少为两个，状态也只保留一个，和普通循环神经网络保持一致。

序列到序列网络（Sequence-to-Sequence）
LSTM与其变体很好地解决了网络中梯度爆炸和消失的问题。但LSTM有一个缺陷，无法处理输入和输出序列不等长的问题，为此提出了序列到序列（Sequence-to-Sequence, 简称Seq2Seq）模型，引入和编码解码机制（Encoder-Decoder），在机器翻译等领域取得了很大的成果，进一步提升了循环神经网络的处理范围。

LSTM-长短时记忆（Long Short Term Memory, LSTM）基本原理

循环神经网络（RNN）的提出，**使神经网络可以训练和解决带有时序信息的任务**，大大拓宽了神经网络的使用范围。但是原始的RNN有明显的缺陷。不管是双向RNN，还是深度RNN，都有一个严重的缺陷：**训练过程中经常会出现梯度爆炸和梯度消失的问题，以至于原始的RNN很难处理长距离的依赖。**

从实例角度

例如，在语言生成问题中：

佳佳今天帮助妈妈洗碗，帮助爸爸修理椅子，还帮助爷爷奶奶照顾小狗毛毛，大家都夸奖了 _____。

例句中出现了很多人，空白出要填谁呢？我们知道是“佳佳”，传统RNN无法很好学习这么远距离的依赖关系。

从理论角度

根据循环神经网络的反向传播算法，可以得到任意时刻k, 误差项沿时间反向传播的公式如下： $$ \delta^T_k=\delta^T_t \prod_{i=k}^{t-1} diag[f'(z_i)]W $$

其中 f为激活函数，z_i为神经网络在第i时刻的加权输入， W为权重矩阵，$diag$表示一个对角矩阵。

注意，由于使用链式求导法则，式中有一个连乘项 $\prod_{i=k}^{t-1} diag[f'(z_i)]W$, 如果激活函数是挤压型，例如 $Tanh$ 或 $sigmoid$, 他们的导数值在 [0,1] 之间。我们再来看W。 1. 如果W的值在 (0,1) 的范围内， 则随着t的增大，连乘项会越来越趋近于0， 误差无法传播，这就导致了 **梯度消失** 的问题。 2. 如果W的值很大，使得$diag[f'(z_i)]W$的值大于1， 则随着t的增大，连乘项的值会呈指数增长，并趋向于无穷，产生 **梯度爆炸**。

梯度消失使得误差无法传递到较早的时刻，权重无法更新，网络停止学习。梯度爆炸又会使网络不稳定，梯度过大，权重变化太大，无法很好学习，最坏情况还会产生溢出（NaN）错误而无法更新权重。

解决办法

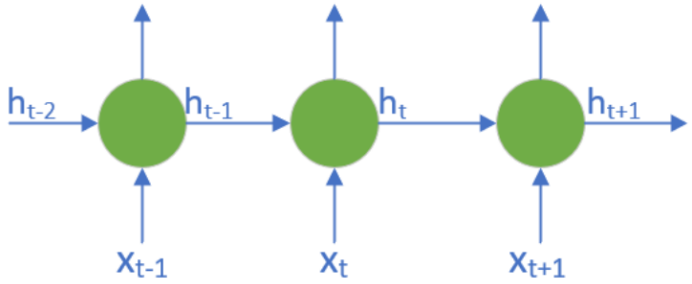
为了解决这个问题，科学家们想了很多办法。

1. 采用半线性激活函数ReLU代替 挤压型激活函数，ReLU函数在定义域大于0的部分，导数恒等于1，来解决梯度消失问题。
2. 合理初始化权重W，使$diag[f'(z_i)]W$的值尽量趋近于1，避免梯度消失和梯度爆炸。

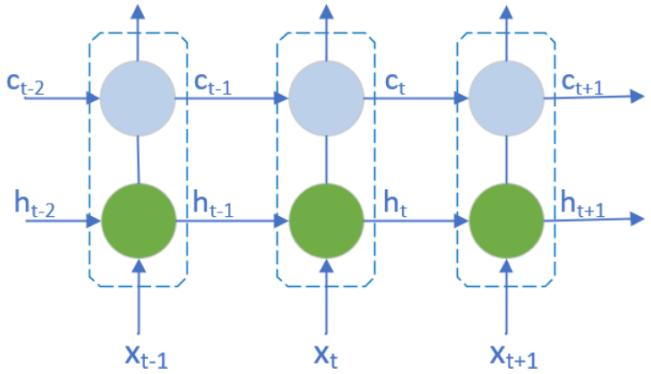
上面两种办法都有一定的缺陷，ReLU函数有自身的缺点，而初始化权重的策略也抵不过连乘操作带来的指数增长问题。要想根本解决问题，必须去掉连乘项。

LSTM网络

LSTM 的设计思路比较简单，原来的RNN中隐藏层只有一个状态h，对短期输入敏感，现在再增加一个状态c，来保存长期状态。这个新增状态称为 细胞状态（cell state）**或**单元状态。



传统RNN结构示意图



LSTM结构示意图

如何控制长期状态c呢？在任意时刻t，我们需要确定三件事：

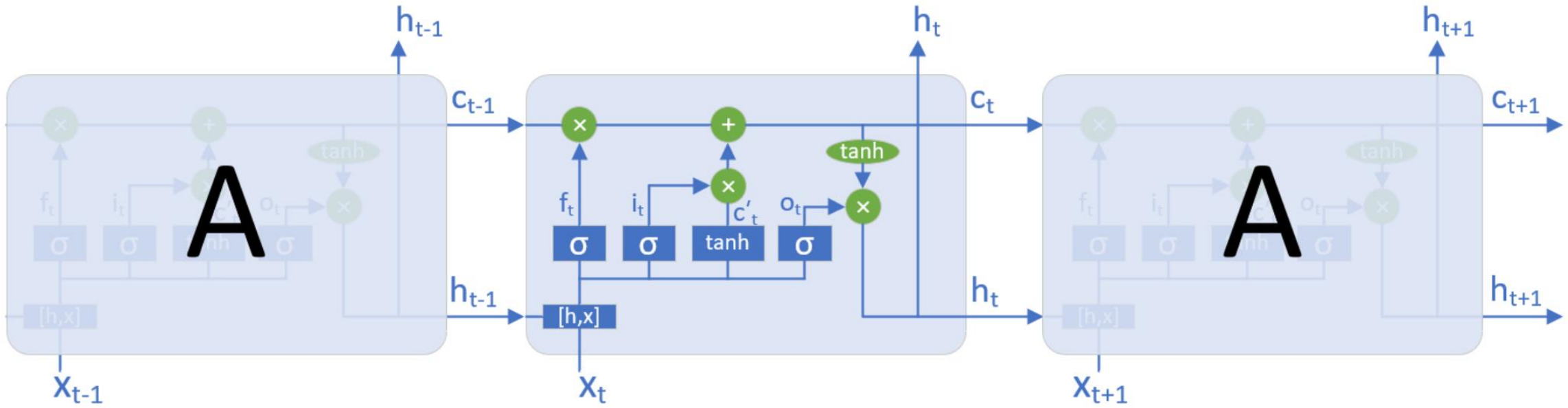
1.t−1时刻传入的状态ct−1，有多少需要保留。

2.当前时刻的输入信息，有多少需要传递到t+1时刻。

3.当前时刻的隐层输出ht是什么。

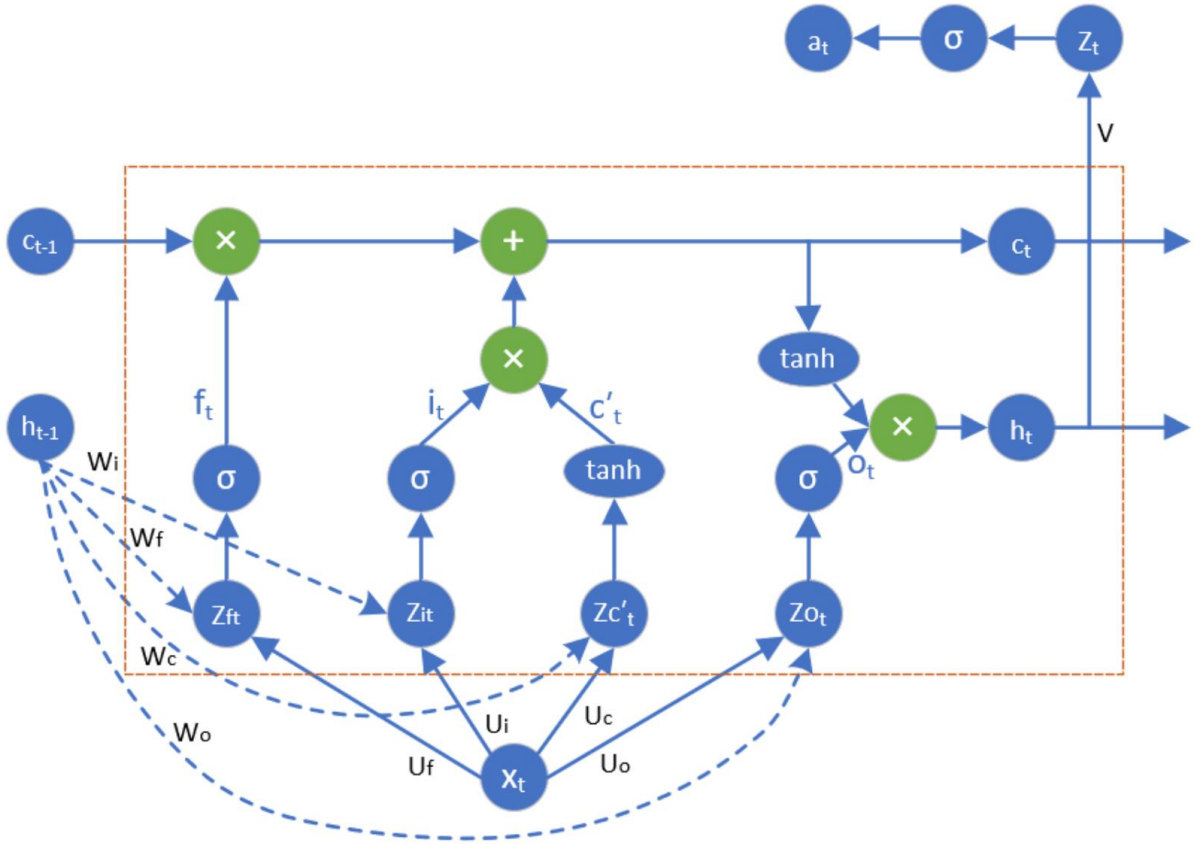
LSTM网络

LSTM设计了 门控（gate） 结构，控制信息的保留和丢弃。LSTM有三个门，分别是：遗忘门（forget gate），输入门（input gate）和输出门（output gate）。



LSTM的反向传播

LSTM使用时序反向传播算法（Backpropagation Through Time, BPTT）进行计算。下图是带有一个输出的LSTM cell。我们使用该图来推导反向传播过程。



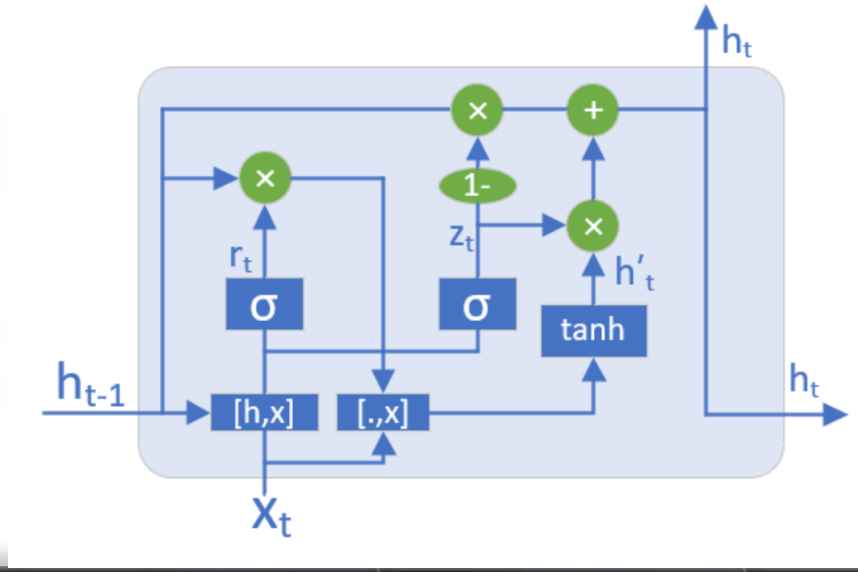
GRU基本原理与实现

LSTM 存在很多变体，其中门控循环单元（Gated Recurrent Unit, GRU）是最常见的一种，也是目前比较流行的一种。
GRU是由 Cho 等人在2014年提出的，它对LSTM做了一些简化：

1.GRU将LSTM原来的三个门简化成为两个：重置门 rt（Reset Gate）和更新门 zt (Update Gate)。

2.GRU不保留单元状态 ct，只保留隐藏状态 ht作为单元输出，这样就和传统RNN的结构保持一致。

3.重置门直接作用于前一时刻的隐藏状态 ht−1。



1. 更新门

$$z_t = \sigma(h_{t-1} \cdot W_z + x_t \cdot U_z)$$

2. 重置门

$$r_t = \sigma(h_{t-1} \cdot W_r + x_t \cdot U_r)$$

3. 候选隐藏状态

$$\tilde{h}_t = \tanh((r_t \circ h_{t-1}) \cdot W_h + x_t \cdot U_h)$$

4. 隐藏状态

$$h = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t$$

GRU的原理浅析

从上面的公式可以看出，GRU通过更新们和重置门控制长期状态的遗忘和保留，以及当前输入信息的选择。更新门和重置门通过$sigmoid$函数，将输入信息映射到$[0,1]$区间，实现门控功能。

首先，上一时刻的状态h_{t-1}通过重置门，加上当前时刻输入信息，共同构成当前时刻的即时状态$\tilde{h}_t$，并通过tanh函数映射到$[-1,1]$区间。

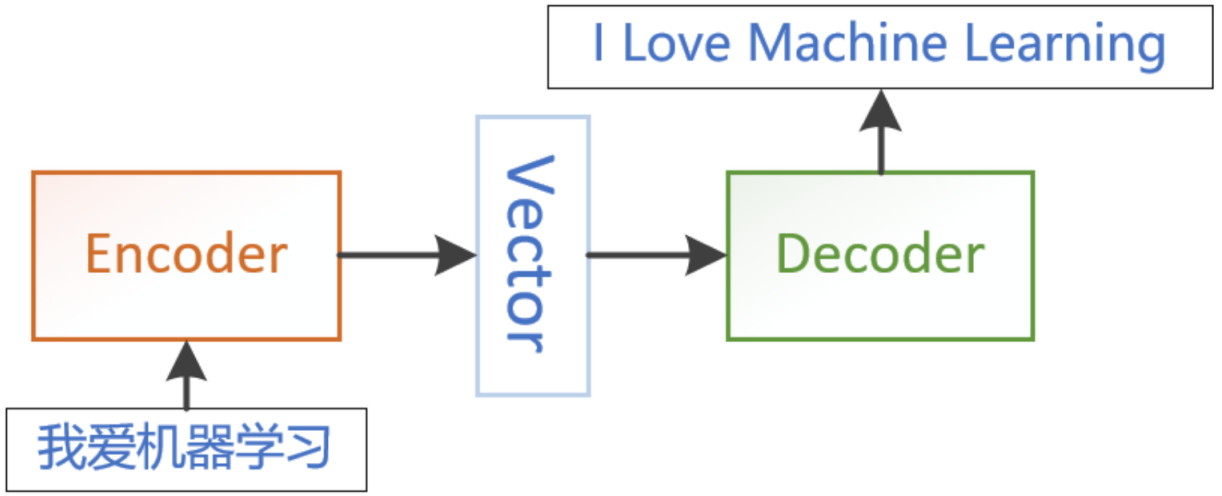
然后，通过更新门实现遗忘和记忆两个部分。从隐藏状态的公式可以看出，通过z_t进行选择性的遗忘和记忆。$(1-z_t)$和z_t有联动关系，上一时刻信息遗忘的越多，当前信息记住的就越多，实现了LSTM中f_t和i_t的功能。

序列到序列

序列到序列模型在自然语言处理中应用广泛，是重要的模型结构。

序列到序列问题有很多应用场景：比如机器翻译、问答系统（QA）、文档摘要生成等。简单的RNN或LSRM结构无法处理这类问题，于是科学家们提出了一种新的结构 —— 编码解码（Encoder-Decoder）结构。

编码-解码结构（Encoder-Decoder）



Encoder-Decoder结构的处理流程非常简单直观。

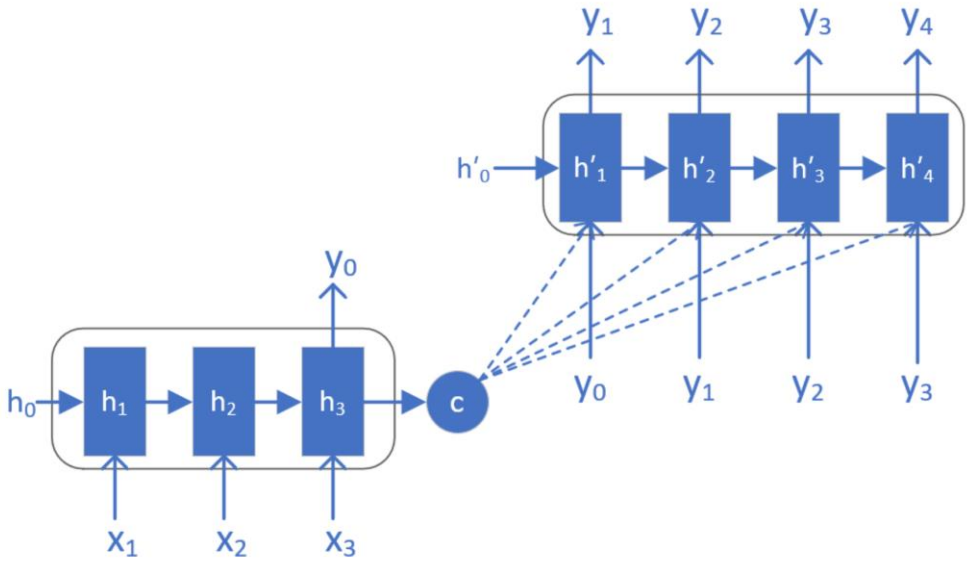
•示意图中，输入序列和输出序列分别为中文语句和翻译之后的英文语句，它们的长度不一定相同。通常会将输入序列嵌入（Embedding）成一定维度的向量，传入编码器。

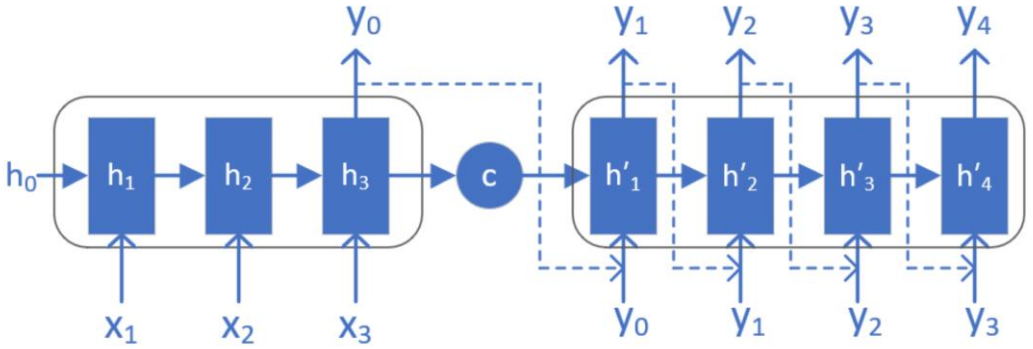
•Encoder为编码器，将输入序列编码成为固定长度的状态向量，通常称为语义编码向量。

•Decoder为解码器，将语义编码向量作为原始输入，解码成所需要的输出序列。

在具体实现中，编码器、解码器可以有不同选择，可自由组合。常见的选择有CNN、RNN、GRU、LSTM等。
应用Encoder-Decoder结构，可构建出序列到序列模型

序列到序列模型（Seq2Seq）





两种结构的编码过程完全一致。

序列到序列模型（Seq2Seq）

两种结构的编码过程完全一致。

输入序列为 $x = [x1, x2, x3]$。

RNN网络中，每个时间节点隐层状态为:

$$h_t = f(h_{t-1}, x_t), \quad t \in [1, 3]$$

编码器中输出的语义编码向量可以有三种不同选取方式，分别是:

$$c = h_3$$

$$c = g(h_3)$$

$$c = g(h1, h2, h3)$$

解码过程

两种结构解码过程的不同点在于，语义编码向量是否应用于每一时刻输入。

第一种结构，每一时刻的输出y_t由前一时刻的输出y_{t-1}、前一时刻的隐层状态h'_{t-1}和c共同决定，即： $y_t = f(y_{t-1}, h'_{t-1}, c)$。

第二种结构，c只作为初始状态传入解码器，并不参与每一时刻的输入，即:

$$\begin{cases} y_1 = f(y_0, h'_0, c) \\ y_t = f(y_{t-1}, h'_{t-1}), t \in [2, 4] \end{cases}$$

以上是序列到序列模型的结构简介，具体实现将在以后补充。

案例

QnA

Microsoft

Reactor

Thank You!