# 激活函数

近年来，深度学习在计算机视觉领域取得了引人注目的成果，其中一个重要因素是激活函数的发展。新型激活函数ReLU克服了梯度消失，使得深度网络的直接监督式训练成为可能。

在人工神经网络中，**神经元节点的激活函数定义了对神经元输出的映射**，简单来说，神经元的输出（例如，全连接网络中就是输入向量与权重向量的内积再加上偏置项）经过激活函数处理后再作为输出。神经网络中激活函数的主要作用是提供网络的**非线性建模能力**，如不特别说明，激活函数一般而言是非线性函数。对于无非线性激励函数情况，每一层网络输出都为线性函数，可验证无论神经网络层数多少，输出都是输入的线性组合，意味着单层网络也可实现，这也是最原始的感知机。引入非线性激励函数，使得深层神经网络变得有意义，也能模拟更复杂的模型。(**为什么引入激活函数可以带来非线性能力？？？**)

虽然2006年Hinton教授提出通过分层无监督预训练解决深层网络训练困难的问题，但是深度网络的直接监督式训练的最终突破，最主要的原因是采用了新型激活函数ReLU[5, 6]。与传统的sigmoid激活函数相比，ReLU能够有效缓解梯度消失问题，从而直接以监督的方式训练深度神经网络，无需依赖无监督的逐层预训练。ReLU 在x<0 时硬饱和。由于 x>0时导数为 1，所以，ReLU 能够在x>0时保持梯度不衰减，从而缓解梯度消失问题。但随着训练的推进，部分输入会落入硬饱和区，导致对应权重无法更新。这种现象被称为“神经元死亡”。

传统预训练的用处：规则化，防止过拟合；压缩数据，去除冗余；强化特征，减小误差；加快收敛速度。标准的sigmoid输出不具备稀疏性，需要用一些惩罚因子来训练出一大堆接近0的冗余数据来，从而产生稀疏数据，例如L1、L1/L2或Student-t作惩罚因子。因此需要进行无监督的预训练。而ReLU是线性修正，公式为：g(x) = max(0, x)，是purelin的折线版。它的作用是如果计算出的值小于0，就让它等于0，否则保持原来的值不变。这是一种简单粗暴地强制某些数据为0的方法，然而经实践证明，训练后的网络完全具备适度的稀疏性。而且训练后的可视化效果和传统方式预训练出的效果很相似，这也说明了ReLU具备引导适度稀疏的能力。

# RNNs

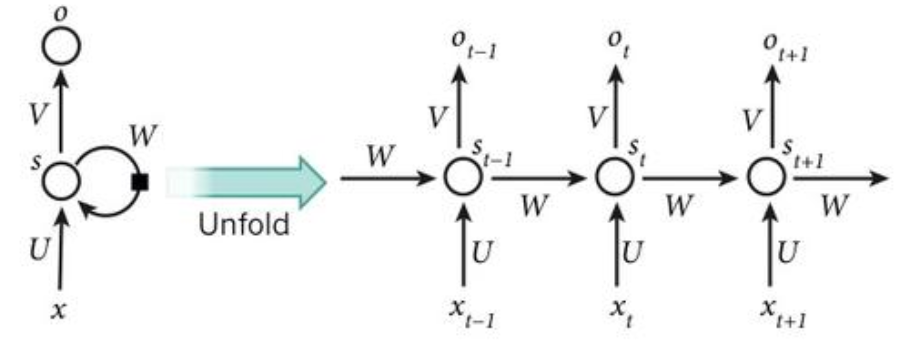
对于涉及到序列输入的任务，比如语音和语言，利用RNNs能获得更好的效果。RNNs一次处理一个输入序列元素，同时维护网络中隐式单元中隐式的包含过去时刻序列元素的历史信息的“状态向量”。

RNNs被发现可以很好的预测文本中下一个字符或者句子中下一个单词，并且可以应用于更加复杂的任务。

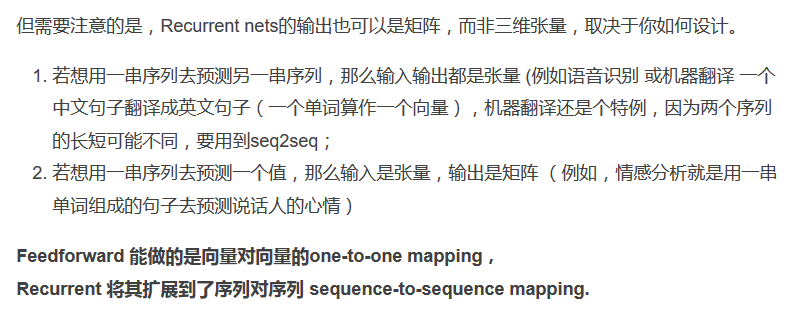
**应用-翻译：**例如在某时刻阅读英语句子中的单词后，将会训练一个英语的“编码器”网络，使得隐式单元的最终状态向量能够很好地表征句子所要表达的意思或思想。这种“思想向量”（thought vector）可以作为联合训练一个法语“编码器”网络的初始化隐式状态（或者额外的输入），其输出为法语翻译首单词的概率分布。如果从分布中选择一个特殊的首单词作为编码网络的输入，将会输出翻译的句子中第二个单词的概率分布，并直到停止选择为止。总体而言，这一过程是根据英语句子的概率分布而产生的法语词汇序列。这种简单的机器翻译方法的表现甚至可以和最先进的（state-of-the-art）的方法相媲美，同时也引起了人们对于理解句子是否需要像使用推理规则操作内部符号表示质疑。这与日常推理中同时涉及到根据合理结论类推的观点是匹配的。

**应用-描述图片：**类比于将法语句子的意思翻译成英语句子，同样可以学习将图片内容“翻译”为英语句子（如图3）。这种编码器是可以在最后的隐层将像素转换为活动向量的深度卷积网络（ConvNet）。解码器与RNNs用于机器翻译和神经网络语言模型的类似。

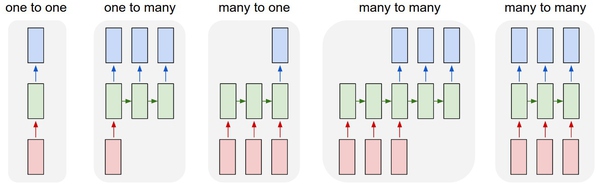
**缺陷：**RNNs是非常强大的动态系统，但是训练它们被证实存在问题的，因为反向传播的梯度在每个时间间隔内是增长或下降的，所以经过一段时间后将导致结果的激增或者降为零。RNNs一旦展开（如图5），可以将之视为一个所有层共享同样权值的深度前馈神经网络。虽然它们的目的是学习长期的依赖性，但理论的和经验的证据表明很难学习并长期保存信息。



## RNNs更详细的解释



**RNN的思想**是: current output不仅仅取决于current input，还取决于previous state；可以理解成**current output是由current input和previous hidden state两个输入计算而出的**。并且每次计算后都会有信息残留于previous hidden state中供下一次计算。如图：



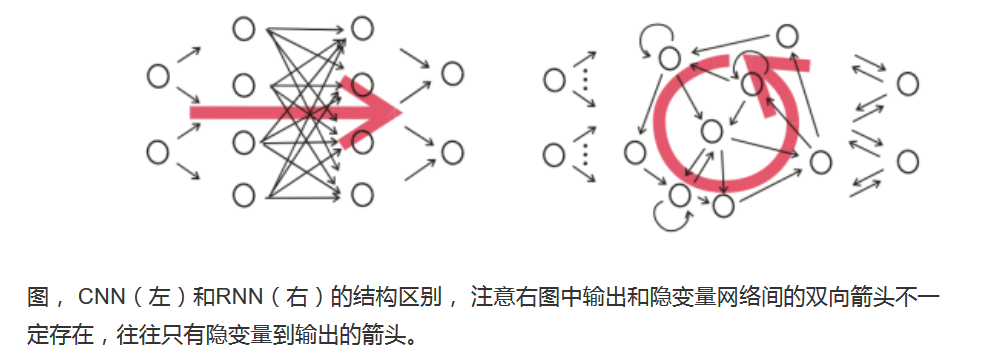
(注：图中最左侧的one to one是feedforward 能做的，右侧都是Recurrent所扩展的)

图2到5是RNN的几种基本玩法。图2是把单一输入转化为序列输出，例如把图像转化成一行文字。 图三是把序列输入转化为单个输出， 比如情感测试，测量一段话正面或负面的情绪。 图四是把序列转化为序列， 最典型的是机器翻译， 注意输入和输出的“时差”。 图5是无时差的序列到序列转化， 比如给一个录像中的每一帧贴标签。

可以将Recurrent的横向操作视为**累积已发生的事情**，并且LSTM的memory cell机制会选择**记忆**或者**忘记**所累积的信息来预测某个时刻的输出。

“循环”两个字，已经点出了RNN的核心特征， 即系统的输出会保留在网络里，和系统下一刻的输入一起共同决定下一刻的输出。**这就把动力学的本质体现了出来， 循环正对应动力学系统的反馈概念，可以刻画复杂的历史依赖。另一个角度看也符合著名的图灵机原理。** **即此刻的状态包含上一刻的历史，又是下一刻变化的依据。** 这其实包含了可编程神经网络的核心概念，即， 当你有一个未知的过程，但你可以测量到输入和输出， **你假设当这个过程通过RNN的时候，它是可以自己学会这样的输入输出规律的， 而且因此具有预测能力。**

## 与CNN的关联



CNN是不是也可以当做RNN来用呢? 答案是否定的，R**NN的重要特性是可以处理不定长的输入，得到一定的输出**。当你的输入可长可短， 比如训练翻译模型的时候， 你的句子长度都不固定，你是无法像一个训练固定像素的图像那样用CNN搞定的。而利用RNN的循环特性可以轻松搞定。

# LTSM

为了解决这个问题，一个增大网络存储的想法随之产生。采用了特殊隐式单元的LSTM（long short-termmemory networks）被首先提出，其自然行为便是长期的保存输入。一种称作记忆细胞的特殊单元**类似累加器和门控神经元**：**它在下一个时间步长将拥有一个权值并联接到自身，拷贝自身状态的真实值和累积的外部信号，但这种自联接是由另一个单元学习并决定何时清除记忆内容的乘法门控制的。**

# 正则化

1. **L0范数与L1范数(提供稀疏)**

L0范数是指向量中非0的元素的个数

L1范数是指向量中各个元素绝对值之和，也有个美称叫“稀疏规则算子”（Lasso regularization）。

1. L2范数

||W||2。它的强大功效是改善机器学习里面一个非常重要的问题：过拟合。

 L2范数是指向量各元素的平方和然后求平方根。我们让L2范数的规则项||W||2最小，可以使得W的每个元素都很小，都接近于0，但与L1范数不同，它不会让它等于0，而是接近于0，这里是有很大的区别的哦。而越小的参数说明模型越简单，越简单的模型则越不容易产生过拟合现象。