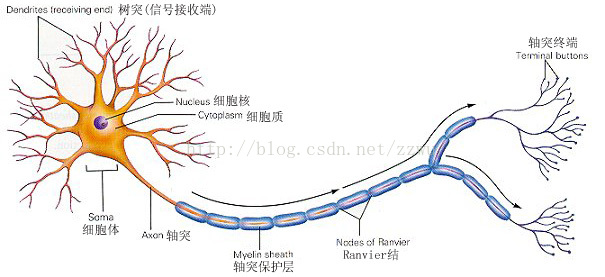
神经网络

人工神经网络（Artificial Neural Networks，简写为ANNs）也简称为神经网络（NNs）或称作连接模型（Connection Model），它是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。在机器学习领域也将人工神经网络简称为神经网络（Neural Network）其而现在神经网络已经广泛的使用在机器学习和模式识别的任务中。取得了令人意向不到效果。

神经网络中最基本的组成单元是神经元（Neuron）模型。该结构来源于生物学中的神经元的结构的研究。身体神经元如下图所示。在人的生命的最初9个月内，这些细胞以每分钟25,000个的惊人速度被创建出来。神经细胞和人身上任何其他类型细胞十分不同，每个神经细胞都长着一根像电线一样的称为轴突（axon）的东西，它的长度有时伸展到几厘米，用来将信号传递给其他的神经细胞。神经细胞的结构如图1 所示。它由一个细胞体(soma)、一些树突(dendrite) 、和一根可以很长的轴突组成。神经细胞体是一颗星状球形物，里面有一个核(nucleus)。树突由细胞体向各个方向长出，本身可有分支，是用来接收信号的。轴突也有许多的分支。轴突通过分支的末梢(terminal)和其他神经细胞的树突相接触,形成所谓的突触(Synapse)，一个神经细胞通过轴突和突触把产生的信号送到其他的神经细胞。神经细胞利用电-化学过程交换信号。输入信号来自另一些神经细胞。这些神经细胞的轴突末梢（也就是终端）和本神经细胞的树突相遇形成突触（synapse），信号就从树突上的突触进入本细胞。信号在大脑中实际怎样传输是一个相当复杂的过程，但就我们而言，重要的是把它看成和现代的计算机一样，利用一系列的0和1来进行操作。就是说，大脑的神经细胞也只有两种状态：兴奋（fire）和不兴奋（即抑制）。发射信号的强度不变，变化的仅仅是频率。神经细胞利用一种我们还不知道的方法,把所有从树突突触上进来的信号进行相加，如果全部信号的总和超过某个阀值，就会激发神经细胞进入兴奋（fire）状态，这时就会有一个电信号通过轴突发送出去给其他神经细胞。如果信号总和没有达到阀值，神经细胞就不会兴奋起来。这样的解释有点过分简单化，但已能满足我们的目的。



正是由于数量巨大的连接，使得大脑具备难以置信的能力。尽管每一个神经细胞仅仅工作于大约100Hz的频率，但因各个神经细胞都以独立处理单元的形式并行工作着，使人类的大脑具有如下功能：

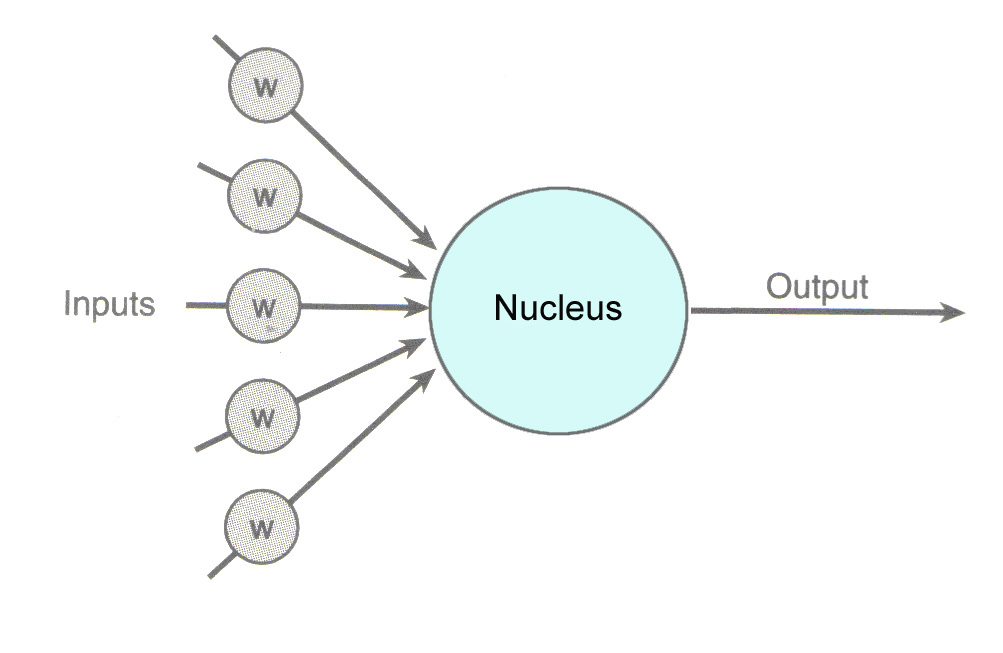
能实现无监督的学习：能够自己进行学习，而不需要导师的监督教导。如果一个神经细胞在一段时间内受到高频率的刺激，则它和输入信号的神经细胞之间的连接强度就会按某种过程改变，使得该神经细胞下一次受到激励时更容易兴奋。从而实现对这种刺激的响应。

对损伤有冗余性：脑即使有很大一部分受到了损伤，它仍然能够执行复杂的工作。对损伤有一定冗余能力。

处理信息的效率极高：神经细胞之间电-化学信号的传递，与一台数字计算机中CPU的数据传输相比，速度是非常慢的，但因神经细胞采用了并行的工作方式，使得大脑能够同时处理大量的数据。例如，大脑视觉皮层在处理通过我们的视网膜输入的一幅图象信号时，大约只要100ms的时间就能完成。考虑到你的神经细胞的平均工作频率只有100Hz，100ms的时间就意味只能完成10个计算步骤，但人们用它却可以正常的生活。

善于归纳推广：大脑和数字计算机不同，它极擅长的事情之一就是模式识别，并能根据已熟悉信息进行归纳推广(generlize)。例如，我们能够阅读他人所写的手稿上的文字，即使我们以前从来没见过他所写的东西。

既然神经网络具有如此之多的性能，那么我们能不能用它来进行机器学习呢？1943年， MCCulloch and Pitts将人体神经元的结构用如下模型进行模拟。而这个模型也是现今人工神经网络的基本组成单元，称之为神经元（Neuron）。现阶段的人工神经网络



在这个神经网络单元中，神经元接收到来自n个其他神经元传递过来的输入信号，这些输入信号通过带权值的连接进行传递，神经元接收到的总输入将与神经元的阈值进行比较，然后通过激活函数的处理产生神经元的输出。用数学函数来表示这个过程如下：



其中表示的是当前神经元的输入信号，一般来说它是模型的输入变量或者是其他神经元的输出，表示的是输入信号对当前神经元的权重。表示的是当前输入信号的线性加权和，即全部的输入信号，a表示输入信号u经过激活函数f()产生的响应，通过这个激活信号是我们要输出的目标函数值或者其他神经元的输入信号。从公式看，这个模型非常的简单，但却是可以实现多各种函数的拟合，实现对目标的分类或者回归。这要是因为两点：1.该模型由许许多多的神经元过程；2.激活函数是非线性函数。神经网络就是通过大量的神经单元和非线性激活实现的超级强大的机器学习的性能。

下图现实目前常用的激活函数和其求得偏微分的结果：

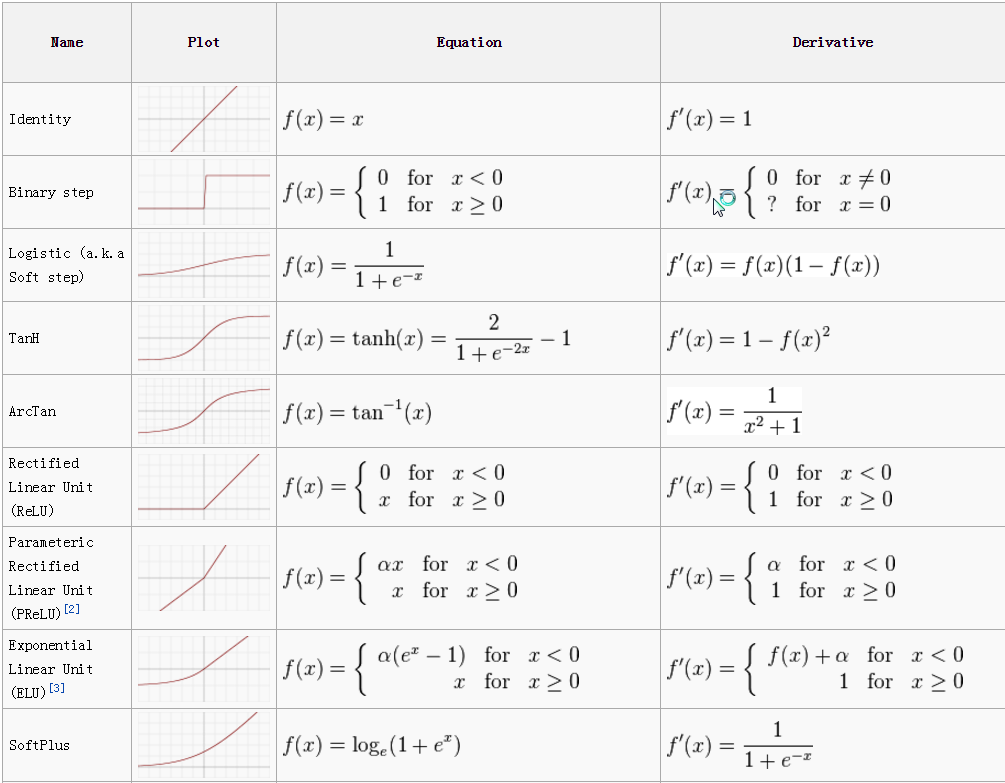


表 1

**下面其中的部分激活函数进行介绍：**

**Identity函数**：将输入信号直接作为激活响应，在神经网络中应用的很少，主要是在某些时候将信号传递过去。

Binary step函数：主要

**Sigmoid函数：**可以看到Sigmoid函数在x想着正无穷和负无穷方向的取值逐渐接近+1和-1，从而使Sigmoid函数可以对任意响应实现激活操作，而不会出现激活过大的情况。而当输入的信号接近0的时候，可以看到Sigmoid函数在0附近呈现出类似线性函数f(x)=(x+0.5)的性质，从而致使在0附近的信号可以几乎全部通过该激活函数从而进入后面的神经元。不过。当某个输入信号过大会使得其它输入信号无法对激活函数起作用。利用在中，某一个输入的信号非常大，则最后u的会非常大，从而导致激活后的响应a非常接近于1，这会时的以后无论其他输入信号如何改变，该神经元的激励响应都不会改变。一个不会改变的神经元在神经网络中是没有作用的。再者，当u非常大时，也会使该函数的偏倒数很小，而神经网络在参数学习的过程中，需要进行误差反向传播，而倒数很小如导致反向传播的误差降低，从而影响网络学习的效果。这一点我们将在后面网络参数学习的过程中进行进一步的介绍。

**Tanh函数：**这个函数在远端（即x趋向于正、负无穷）时和Sigmoid函数具有相似的性质，而在x=0处则可以使得实现f(x)=x的近似估计，从而实现非正激励函数的传输。

**ReLU/PReLU函数**：主要是对0两侧的信号值做不同的处理，在x>0时，是一个线性函数，而在x<0时则是0或者不同参数的线性函数。相比于Sigmoid函数或者Tanh函数，这两个函数在计算激活的值或者求其倒数时更加的简单。不过可以看到这两个函数当x无穷大时的输出也是无穷大的，因此无法抑制激活值的范围，不过从另一个方面可以看到，无论x多大这两个激活函数的倒数都是一个定值，这确保了在参数求解的反向求解过程中残差的回溯。

**浅层神经网络**

**描述的是一个目前研究最为成熟Shallow 结构的神经网络（只含有单层隐藏层神经元的结构）。第一层为输入层 (input layer )，第二层称为隐藏层 ( hidden layer )，最后一层为输出层( output layer )。神经元之间都是由低层出发，终止于高层神经元的一条有向边进行连接，每条边都有自己的权重。每个神经元都是一个计算单元，如在Feed-forward neural network 中，除输入层神经元外，每个神经元为一个计算单元，可以通过一个计算函数 f() 来表示，函数的具体形式可以自己定义，现在用的较多的是 感知器计算神经元，如果你对感知器有所了解的话，理解起来会容易很多。 可以计算此时神经元所具有的能量值，当该值超过一定阀值的时候神经元的状态就会发生改变，神经元只有两种状态，激活或未激活。在实际的人工神经网络中，一般是用一种概率的方式去表示神经元是否处于激活状态，可以用 h(f) 来表示，f 代表神经元的能量值，h(f) 代表该能量值使得神经元的状态发生改变的概率有多大，能量值越大，处于激活状态的概率就越高。到这部分你已经接触到了关于神经网络的几个基本术语，下面用更加规范的符号来表示，神经元的激活值(activations) f() ，表示计算神经元的能量值, 神经元的激活状态 h(f) ，h 表示激活函数。**



**我们用表示输入层的输入特征，用表示第l层神经元，用表示隐含层某个节点的输入特征的权重和偏至。隐含层神经元的输入用u表示，隐含层输出的激活信号用a表示，则有某一个神经元的输出为：**

** （1）**

**其中函数f()为表（1）中列举的一些非线性函数。一般隐含层的输出a可以作为其他隐含层的输入或者作为最后的结果输出。在这个样例中，我们用表示隐含层的输出。输出层的输出为：**

** （2）**

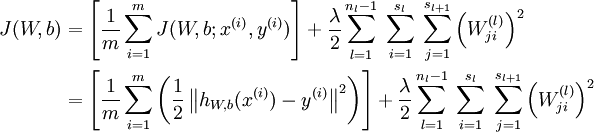
**其中将作为最终的输出函数。**

**优化目标函数**

**一般利用神经网络的输出和问题的目标构成损失函数（Loss Function）。在上面的示例中我们假定有m个样本，目标为对变量y的回归，即通过特征向量x估计目标y的值。这里采用的是二次平方和最小化 (最小二乘)作为损失函数，用该公式表示损失函数而在一般的优化问题中，往往对网络模型中的权重加正则化约束，则最后的目标函数为**

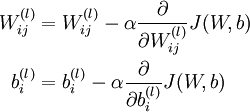


（3）

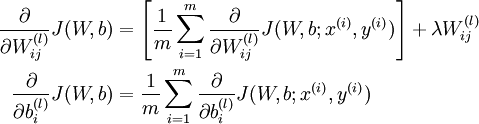


对上述目标往往利用梯度下降法进行求解。在整个优化的过程中，最重要的就是如何求解不同W^{(l)}\_{ij}的梯度信息，其中一个重要的算法就是误差反向传播(error backpropagation)学习算法。假设我们可以求出目标函数对每个参数（权重和偏差）的偏倒数，则参数的更新算法可以用如下函数表示：

（4）



其中



方向传播算法**backpropagation**

1，先执行一次前向传播，计算出每一层的神经元的激活值，直到输出层为止



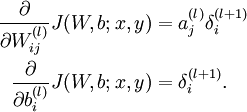
2，对于每一个输出层神经元，采用如下方式计算误差(实际上在Andrew的资料中省略了很多推到部分，该部分我后面会放上，你们先理解个大概过程)



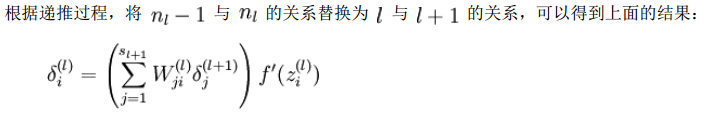
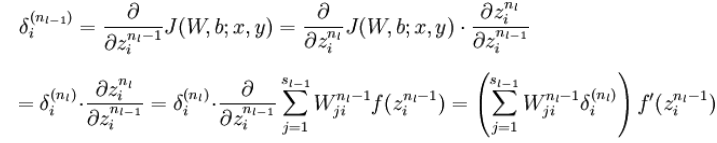
3，对于较低层的神经元的误差用如下公式进行表示，



4，关于不同参数的梯度



下面来看下项的具体推导过程。

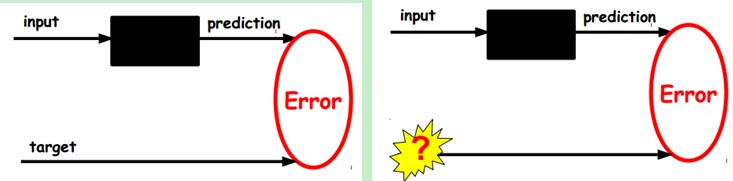


**AutoEncoder自动编码器**

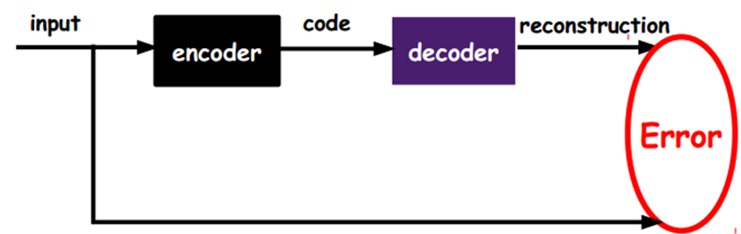
        Deep Learning最简单的一种方法是利用人工神经网络的特点，人工神经网络（ANN）本身就是具有层次结构的系统，如果给定一个神经网络，我们假设其输出与输入是相同的，然后训练调整其参数，得到每一层中的权重。自然地，我们就得到了输入I的几种不同表示（每一层代表一种表示），这些表示就是特征。自动编码器就是一种尽可能复现输入信号的神经网络。为了实现这种复现，自动编码器就必须捕捉可以代表输入数据的最重要的因素，就像PCA那样，找到可以代表原信息的主要成分。

       具体过程简单的说明如下：

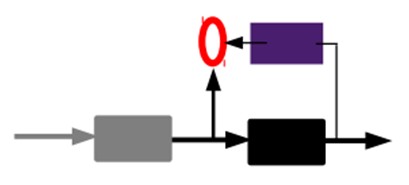
**1）给定无标签数据，用非监督学习学习特征：**



       在我们之前的神经网络中，如第一个图，我们输入的样本是有标签的，即（input, target），这样我们根据当前输出和target（label）之间的差去改变前面各层的参数，直到收敛。但现在我们只有无标签数据，也就是右边的图。那么这个误差怎么得到呢？

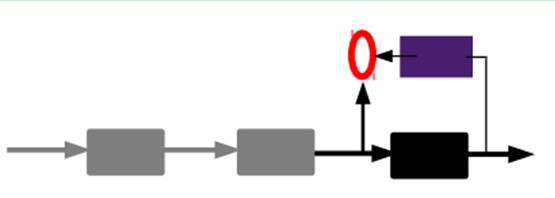


        如上图，我们将input输入一个encoder编码器，就会得到一个code，这个code也就是输入的一个表示，那么我们怎么知道这个code表示的就是input呢？我们加一个decoder解码器，这时候decoder就会输出一个信息，那么如果输出的这个信息和一开始的输入信号input是很像的（理想情况下就是一样的），那很明显，我们就有理由相信这个code是靠谱的。所以，我们就通过调整encoder和decoder的参数，使得重构误差最小，这时候我们就得到了输入input信号的第一个表示了，也就是编码code了。因为是无标签数据，所以误差的来源就是直接重构后与原输入相比得到。



**2）通过编码器产生特征，然后训练下一层。这样逐层训练：**

       那上面我们就得到第一层的code，我们的重构误差最小让我们相信这个code就是原输入信号的良好表达了，或者牵强点说，它和原信号是一模一样的（表达不一样，反映的是一个东西）。那第二层和第一层的训练方式就没有差别了，我们将第一层输出的code当成第二层的输入信号，同样最小化重构误差，就会得到第二层的参数，并且得到第二层输入的code，也就是原输入信息的第二个表达了。其他层就同样的方法炮制就行了（训练这一层，前面层的参数都是固定的，并且他们的decoder已经没用了，都不需要了）。

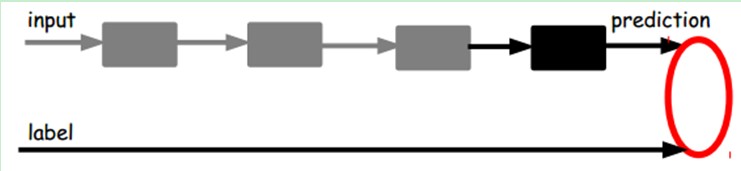


**3）有监督微调：**

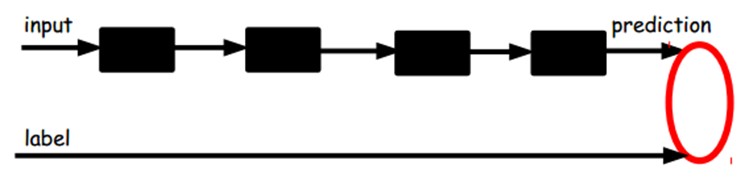
      经过上面的方法，我们就可以得到很多层了。至于需要多少层（或者深度需要多少，这个目前本身就没有一个科学的评价方法）需要自己试验调了。每一层都会得到原始输入的不同的表达。当然了，我们觉得它是越抽象越好了，就像人的视觉系统一样。

       到这里，这个AutoEncoder还不能用来分类数据，因为它还没有学习如何去连结一个输入和一个类。它只是学会了如何去重构或者复现它的输入而已。或者说，它只是学习获得了一个可以良好代表输入的特征，这个特征可以最大程度上代表原输入信号。那么，为了实现分类，我们就可以在AutoEncoder的最顶的编码层添加一个分类器（例如罗杰斯特回归、SVM等），然后通过标准的多层神经网络的监督训练方法（梯度下降法）去训练。

        也就是说，这时候，我们需要将最后层的特征code输入到最后的分类器，通过有标签样本，通过监督学习进行微调，这也分两种，一个是只调整分类器（黑色部分）：



       另一种：通过有标签样本，微调整个系统：（如果有足够多的数据，这个是最好的。end-to-end learning端对端学习）



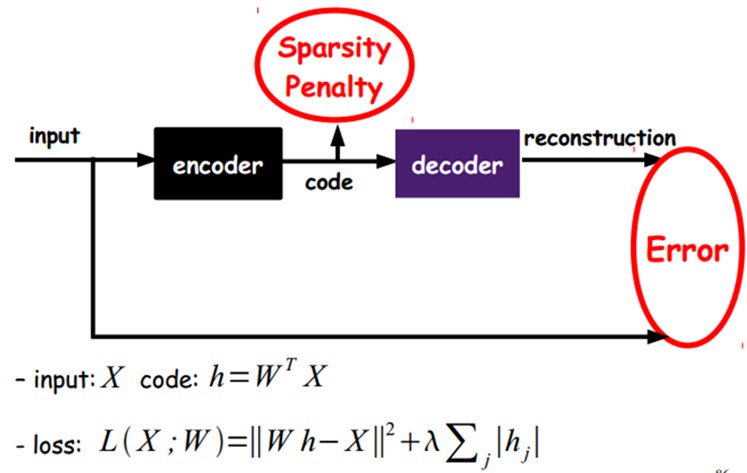
       一旦监督训练完成，这个网络就可以用来分类了。神经网络的最顶层可以作为一个线性分类器，然后我们可以用一个更好性能的分类器去取代它。

       在研究中可以发现，如果在原有的特征中加入这些自动学习得到的特征可以大大提高精确度，甚至在分类问题中比目前最好的分类算法效果还要好！

        AutoEncoder存在一些变体，这里简要介绍下两个：

**Sparse AutoEncoder稀疏自动编码器：**

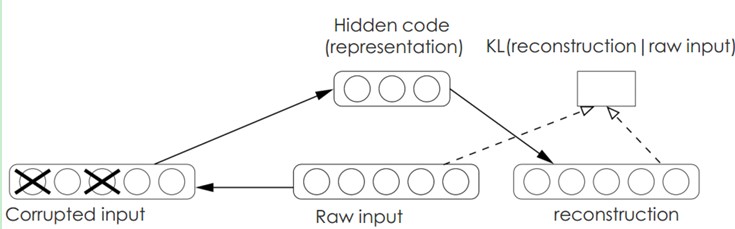
      当然，我们还可以继续加上一些约束条件得到新的Deep Learning方法，如：如果在AutoEncoder的基础上加上L1的Regularity限制（L1主要是约束每一层中的节点中大部分都要为0，只有少数不为0，这就是Sparse名字的来源），我们就可以得到Sparse AutoEncoder法。



       如上图，其实就是限制每次得到的表达code尽量稀疏。因为稀疏的表达往往比其他的表达要有效（人脑好像也是这样的，某个输入只是刺激某些神经元，其他的大部分的神经元是受到抑制的）。

**Denoising AutoEncoders降噪自动编码器：**

        降噪自动编码器DA是在自动编码器的基础上，训练数据加入噪声，所以自动编码器必须学习去去除这种噪声而获得真正的没有被噪声污染过的输入。因此，这就迫使编码器去学习输入信号的更加鲁棒的表达，这也是它的泛化能力比一般编码器强的原因。DA可以通过梯度下降算法去训练。

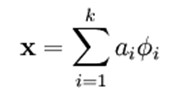


**Sparse Coding稀疏编码**

       如果我们把输出必须和输入相等的限制放松，同时利用线性代数中基的概念，即O = a1\*Φ1 + a2\*Φ2+….+ an\*Φn， Φi是基，ai是系数，我们可以得到这样一个优化问题：

Min |I – O|，其中I表示输入，O表示输出。

       通过求解这个最优化式子，我们可以求得系数ai和基Φi，这些系数和基就是输入的另外一种近似表达。

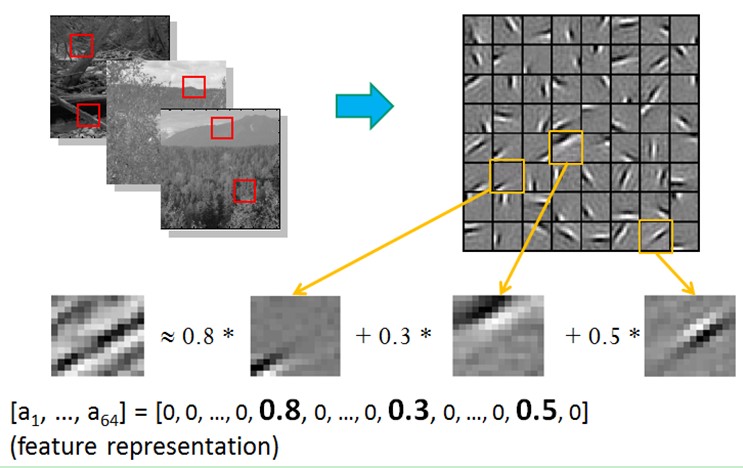


       因此，它们可以用来表达输入I，这个过程也是自动学习得到的。如果我们在上述式子上加上L1的Regularity限制，得到：

Min |I – O| + u\*(|a1| + |a2| + … + |an |)

        这种方法被称为Sparse Coding。通俗的说，就是将一个信号表示为一组基的线性组合，而且要求只需要较少的几个基就可以将信号表示出来。“稀疏性”定义为：只有很少的几个非零元素或只有很少的几个远大于零的元素。要求系数 ai 是稀疏的意思就是说：对于一组输入向量，我们只想有尽可能少的几个系数远大于零。选择使用具有稀疏性的分量来表示我们的输入数据是有原因的，因为绝大多数的感官数据，比如自然图像，可以被表示成少量基本元素的叠加，在图像中这些基本元素可以是面或者线。同时，比如与初级视觉皮层的类比过程也因此得到了提升（人脑有大量的神经元，但对于某些图像或者边缘只有很少的神经元兴奋，其他都处于抑制状态）。

         稀疏编码算法是一种无监督学习方法，它用来寻找一组“超完备”基向量来更高效地表示样本数据。虽然形如主成分分析技术（PCA）能使我们方便地找到一组“完备”基向量，但是这里我们想要做的是找到一组“超完备”基向量来表示输入向量（也就是说，基向量的个数比输入向量的维数要大）。超完备基的好处是它们能更有效地找出隐含在输入数据内部的结构与模式。然而，对于超完备基来说，系数ai不再由输入向量唯一确定。因此，在稀疏编码算法中，我们另加了一个评判标准“稀疏性”来解决因超完备而导致的退化（degeneracy）问题。（[详细过程请参考：UFLDL Tutorial稀疏编码](http://deeplearning.stanford.edu/wiki/index.php/%E7%A8%80%E7%96%8F%E7%BC%96%E7%A0%81)）



       比如在图像的Feature Extraction的最底层要做Edge Detector的生成，那么这里的工作就是从Natural Images中randomly选取一些小patch，通过这些patch生成能够描述他们的“基”，也就是右边的8\*8=64个basis组成的basis，然后给定一个test patch, 我们可以按照上面的式子通过basis的线性组合得到，而sparse matrix就是a，下图中的a中有64个维度，其中非零项只有3个，故称“sparse”。

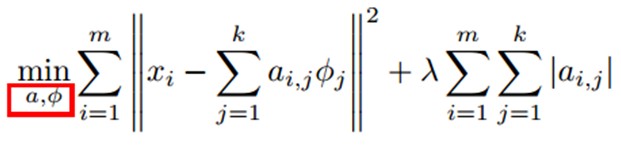
       这里可能大家会有疑问，为什么把底层作为Edge Detector呢？上层又是什么呢？这里做个简单解释大家就会明白，之所以是Edge Detector是因为不同方向的Edge就能够描述出整幅图像，所以不同方向的Edge自然就是图像的basis了……而上一层的basis组合的结果，上上层又是上一层的组合basis……（就是上面第四部分的时候咱们说的那样）

       Sparse coding分为两个部分：

**1）Training阶段：**给定一系列的样本图片[x1, x 2, …]，我们需要学习得到一组基[Φ1, Φ2, …]，也就是字典。

       稀疏编码是k-means算法的变体，其训练过程也差不多（EM算法的思想：如果要优化的目标函数包含两个变量，如L(W, B)，那么我们可以先固定W，调整B使得L最小，然后再固定B，调整W使L最小，这样迭代交替，不断将L推向最小值。EM算法可以见我的博客：“[从最大似然到EM算法浅解](http://blog.csdn.net/zouxy09/article/details/8537620)”）。

       训练过程就是一个重复迭代的过程，按上面所说，我们交替的更改a和Φ使得下面这个目标函数最小。



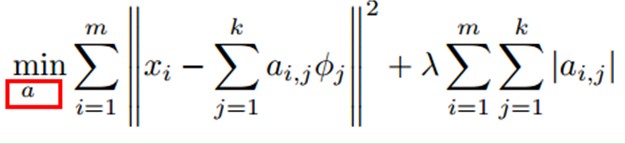
      每次迭代分两步：

a）固定字典Φ[k]，然后调整a[k]，使得上式，即目标函数最小（即解LASSO问题）。

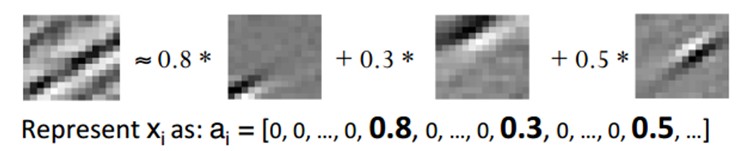
b）然后固定住a [k]，调整Φ [k]，使得上式，即目标函数最小（即解凸QP问题）。

      不断迭代，直至收敛。这样就可以得到一组可以良好表示这一系列x的基，也就是字典。

**2）Coding阶段：**给定一个新的图片x，由上面得到的字典，通过解一个LASSO问题得到稀疏向量**a**。这个稀疏向量就是这个输入向量x的一个稀疏表达了。

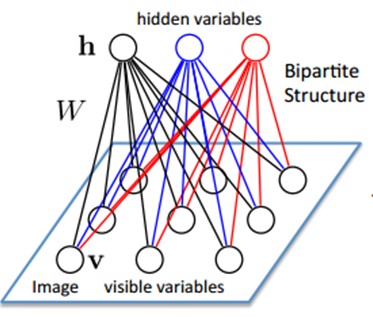


例如：

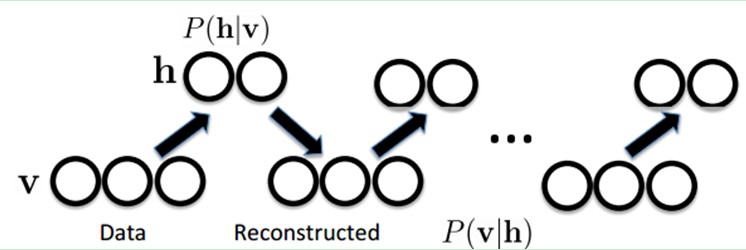


**9.3、Restricted Boltzmann Machine (RBM)限制波尔兹曼机**

       假设有一个二部图，每一层的节点之间没有链接，一层是可视层，即输入数据层（v)，一层是隐藏层(h)，如果假设所有的节点都是随机二值变量节点（只能取0或者1值），同时假设全概率分布p(v,h)满足Boltzmann 分布，我们称这个模型是Restricted BoltzmannMachine (RBM)。

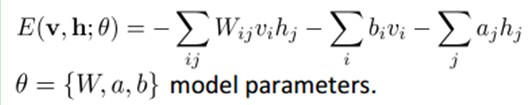


       下面我们来看看为什么它是Deep Learning方法。首先，这个模型因为是二部图，所以在已知v的情况下，所有的隐藏节点之间是条件独立的（因为节点之间不存在连接），即p(h|v)=p(h1|v)…p(hn|v)。同理，在已知隐藏层h的情况下，所有的可视节点都是条件独立的。同时又由于所有的v和h满足Boltzmann 分布，因此，当输入v的时候，通过p(h|v) 可以得到隐藏层h，而得到隐藏层h之后，通过p(v|h)又能得到可视层，通过调整参数，我们就是要使得从隐藏层得到的可视层v1与原来的可视层v如果一样，那么得到的隐藏层就是可视层另外一种表达，因此隐藏层可以作为可视层输入数据的特征，所以它就是一种Deep Learning方法。

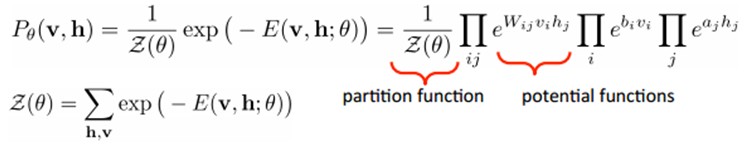


       如何训练呢？也就是可视层节点和隐节点间的权值怎么确定呢？我们需要做一些数学分析。也就是模型了。

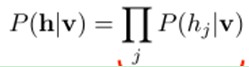
        联合组态（jointconfiguration）的能量可以表示为：



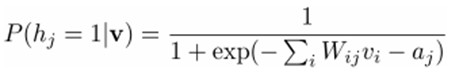
        而某个组态的联合概率分布可以通过Boltzmann 分布（和这个组态的能量）来确定：



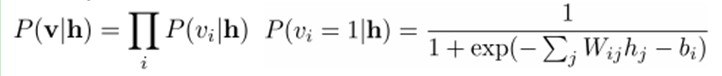
      因为隐藏节点之间是条件独立的（因为节点之间不存在连接），即：



      然后我们可以比较容易（对上式进行因子分解Factorizes）得到在给定可视层v的基础上，隐层第j个节点为1或者为0的概率：

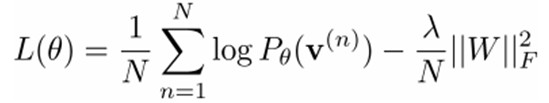


       同理，在给定隐层h的基础上，可视层第i个节点为1或者为0的概率也可以容易得到：

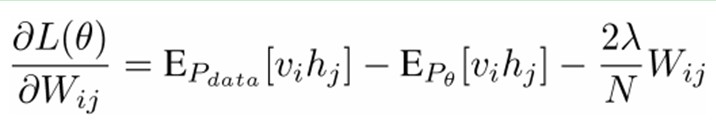


       给定一个满足独立同分布的样本集：D={**v**(1),**v**(2),…,**v**(N)}，我们需要学习参数θ={W,a,b}。

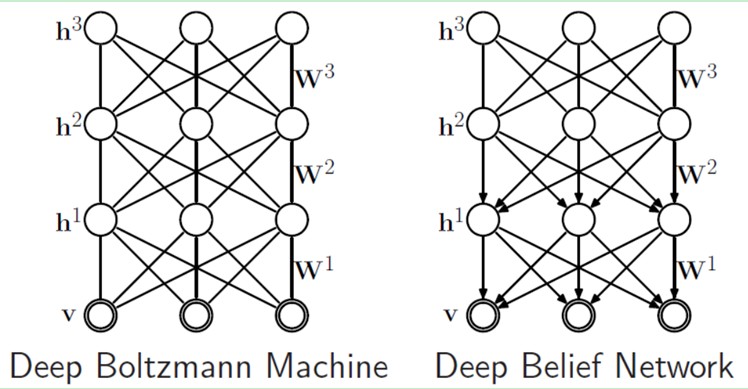
       我们最大化以下对数似然函数（最大似然估计：对于某个概率模型，我们需要选择一个参数，让我们当前的观测样本的概率最大）：



        也就是对最大对数似然函数求导，就可以得到L最大时对应的参数W了。



       如果，我们把隐藏层的层数增加，我们可以得到Deep Boltzmann Machine(DBM)；如果我们在靠近可视层的部分使用贝叶斯信念网络（即有向图模型，当然这里依然限制层中节点之间没有链接），而在最远离可视层的部分使用Restricted Boltzmann Machine，我们可以得到DeepBelief Net（DBN）。



**Convolutional Neural Networks卷积神经网络**

       卷积神经网络是人工神经网络的一种，已成为当前语音分析和图像识别领域的研究热点。它的权值共享网络结构使之更类似于生物神经网络，降低了网络模型的复杂度，减少了权值的数量。该优点在网络的输入是多维图像时表现的更为明显，使图像可以直接作为网络的输入，避免了传统识别算法中复杂的特征提取和数据重建过程。卷积网络是为识别二维形状而特殊设计的一个多层感知器，这种网络结构对平移、比例缩放、倾斜或者共他形式的变形具有高度不变性。

       CNNs是受早期的延时神经网络（TDNN）的影响。延时神经网络通过在时间维度上共享权值降低学习复杂度，适用于语音和时间序列信号的处理。

       CNNs是第一个真正成功训练多层网络结构的学习算法。它利用空间关系减少需要学习的参数数目以提高一般前向BP算法的训练性能。CNNs作为一个深度学习架构提出是为了最小化数据的预处理要求。在CNN中，图像的一小部分（局部感受区域）作为层级结构的最低层的输入，信息再依次传输到不同的层，每层通过一个数字滤波器去获得观测数据的最显著的特征。这个方法能够获取对平移、缩放和旋转不变的观测数据的显著特征，因为图像的局部感受区域允许神经元或者处理单元可以访问到最基础的特征，例如定向边缘或者角点。

**1）卷积神经网络的历史**

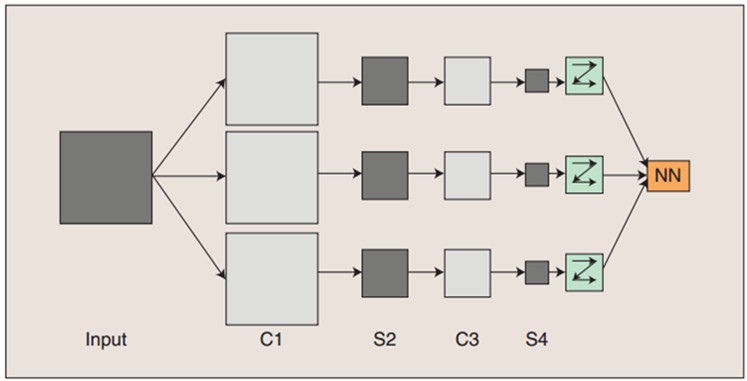
       1962年Hubel和Wiesel通过对猫视觉皮层细胞的研究，提出了感受野(receptive field)的概念，1984年日本学者Fukushima基于感受野概念提出的神经认知机(neocognitron)可以看作是卷积神经网络的第一个实现网络，也是感受野概念在人工神经网络领域的首次应用。神经认知机将一个视觉模式分解成许多子模式（特征），然后进入分层递阶式相连的特征平面进行处理，它试图将视觉系统模型化，使其能够在即使物体有位移或轻微变形的时候，也能完成识别。

       通常神经认知机包含两类神经元，即承担特征抽取的S-元和抗变形的C-元。S-元中涉及两个重要参数，即感受野与阈值参数，前者确定输入连接的数目，后者则控制对特征子模式的反应程度。许多学者一直致力于提高神经认知机的性能的研究：在传统的神经认知机中，每个S-元的感光区中由C-元带来的视觉模糊量呈正态分布。如果感光区的边缘所产生的模糊效果要比中央来得大，S-元将会接受这种非正态模糊所导致的更大的变形容忍性。我们希望得到的是，训练模式与变形刺激模式在感受野的边缘与其中心所产生的效果之间的差异变得越来越大。为了有效地形成这种非正态模糊，Fukushima提出了带双C-元层的改进型神经认知机。

       Van Ooyen和Niehuis为提高神经认知机的区别能力引入了一个新的参数。事实上，该参数作为一种抑制信号，抑制了神经元对重复激励特征的激励。多数神经网络在权值中记忆训练信息。根据Hebb学习规则，某种特征训练的次数越多，在以后的识别过程中就越容易被检测。也有学者将进化计算理论与神经认知机结合，通过减弱对重复性激励特征的训练学习，而使得网络注意那些不同的特征以助于提高区分能力。上述都是神经认知机的发展过程，而卷积神经网络可看作是神经认知机的推广形式，神经认知机是卷积神经网络的一种特例。

**2）卷积神经网络的网络结构**

      卷积神经网络是一个多层的神经网络，每层由多个二维平面组成，而每个平面由多个独立神经元组成。



       图：卷积神经网络的概念示范：输入图像通过和三个可训练的滤波器和可加偏置进行卷积，滤波过程如图一，卷积后在C1层产生三个特征映射图，然后特征映射图中每组的四个像素再进行求和，加权值，加偏置，通过一个Sigmoid函数得到三个S2层的特征映射图。这些映射图再进过滤波得到C3层。这个层级结构再和S2一样产生S4。最终，这些像素值被光栅化，并连接成一个向量输入到传统的神经网络，得到输出。

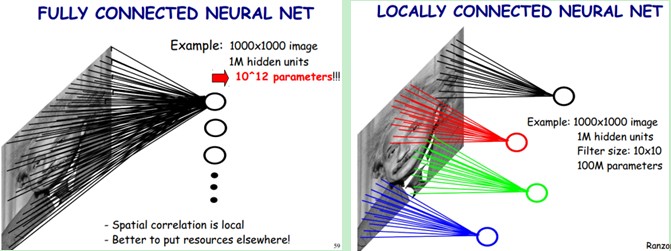
       一般地，C层为特征提取层，每个神经元的输入与前一层的局部感受野相连，并提取该局部的特征，一旦该局部特征被提取后，它与其他特征间的位置关系也随之确定下来；S层是特征映射层，网络的每个计算层由多个特征映射组成，每个特征映射为一个平面，平面上所有神经元的权值相等。特征映射结构采用影响函数核小的sigmoid函数作为卷积网络的激活函数，使得特征映射具有位移不变性。

       此外，由于一个映射面上的神经元共享权值，因而减少了网络自由参数的个数，降低了网络参数选择的复杂度。卷积神经网络中的每一个特征提取层（C-层）都紧跟着一个用来求局部平均与二次提取的计算层（S-层），这种特有的两次特征提取结构使网络在识别时对输入样本有较高的畸变容忍能力。

**3）关于参数减少与权值共享**

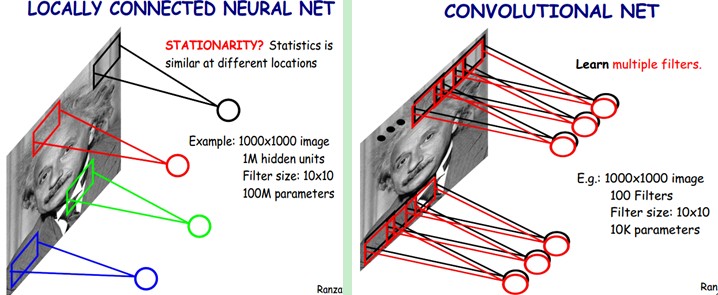
      上面聊到，好像CNN一个牛逼的地方就在于通过感受野和权值共享减少了神经网络需要训练的参数的个数。那究竟是啥的呢？

       下图左：如果我们有1000x1000像素的图像，有1百万个隐层神经元，那么他们全连接的话（每个隐层神经元都连接图像的每一个像素点），就有1000x1000x1000000=10^12个连接，也就是10^12个权值参数。然而图像的空间联系是局部的，就像人是通过一个局部的感受野去感受外界图像一样，每一个神经元都不需要对全局图像做感受，每个神经元只感受局部的图像区域，然后在更高层，将这些感受不同局部的神经元综合起来就可以得到全局的信息了。这样，我们就可以减少连接的数目，也就是减少神经网络需要训练的权值参数的个数了。如下图右：假如局部感受野是10x10，隐层每个感受野只需要和这10x10的局部图像相连接，所以1百万个隐层神经元就只有一亿个连接，即10^8个参数。比原来减少了四个0（数量级），这样训练起来就没那么费力了，但还是感觉很多的啊，那还有啥办法没？

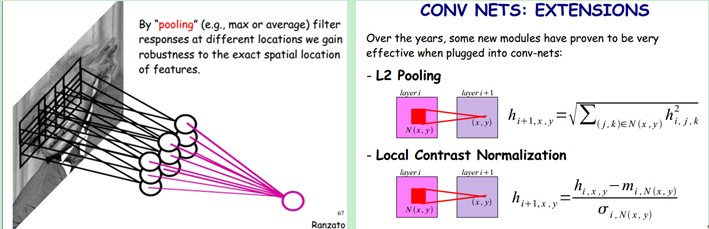


       我们知道，隐含层的每一个神经元都连接10x10个图像区域，也就是说每一个神经元存在10x10=100个连接权值参数。那如果我们每个神经元这100个参数是相同的呢？也就是说每个神经元用的是同一个卷积核去卷积图像。这样我们就只有多少个参数？？只有100个参数啊！！！亲！不管你隐层的神经元个数有多少，两层间的连接我只有100个参数啊！亲！这就是权值共享啊！亲！这就是卷积神经网络的主打卖点啊！亲！（有点烦了，呵呵）也许你会问，这样做靠谱吗？为什么可行呢？这个……共同学习。

       好了，你就会想，这样提取特征也忒不靠谱吧，这样你只提取了一种特征啊？对了，真聪明，我们需要提取多种特征对不？假如一种滤波器，也就是一种卷积核就是提出图像的一种特征，例如某个方向的边缘。那么我们需要提取不同的特征，怎么办，加多几种滤波器不就行了吗？对了。所以假设我们加到100种滤波器，每种滤波器的参数不一样，表示它提出输入图像的不同特征，例如不同的边缘。这样每种滤波器去卷积图像就得到对图像的不同特征的放映，我们称之为Feature Map。所以100种卷积核就有100个Feature Map。这100个Feature Map就组成了一层神经元。到这个时候明了了吧。我们这一层有多少个参数了？100种卷积核x每种卷积核共享100个参数=100x100=10K，也就是1万个参数。才1万个参数啊！亲！（又来了，受不了了！）见下图右：不同的颜色表达不同的滤波器。



       嘿哟，遗漏一个问题了。刚才说隐层的参数个数和隐层的神经元个数无关，只和滤波器的大小和滤波器种类的多少有关。那么隐层的神经元个数怎么确定呢？它和原图像，也就是输入的大小（神经元个数）、滤波器的大小和滤波器在图像中的滑动步长都有关！例如，我的图像是1000x1000像素，而滤波器大小是10x10，假设滤波器没有重叠，也就是步长为10，这样隐层的神经元个数就是(1000x1000 )/ (10x10)=100x100个神经元了，假设步长是8，也就是卷积核会重叠两个像素，那么……我就不算了，思想懂了就好。注意了，这只是一种滤波器，也就是一个Feature Map的神经元个数哦，如果100个Feature Map就是100倍了。由此可见，图像越大，神经元个数和需要训练的权值参数个数的贫富差距就越大。



      需要注意的一点是，上面的讨论都没有考虑每个神经元的偏置部分。所以权值个数需要加1 。这个也是同一种滤波器共享的。

（补充公式）

**循环神经网络（Recurrent Neural Network）**

广为人知的是，给定了一个隐藏节点的集合（可能非常大），传统的前馈网络可以用来近似任何空间受限的有限函数。这就是说，对于拥有确定输入空间的函数，我们总有方法利用神经元网络对这些函数进行编码。对于一个两层的网络，这个映射包含两步。



如果拥有足够的采样点，我们能够利用反向传播等自动学习的技术来寻找网络的权重（G和F）。

回顾神经元网络不仅在输入空间上进行操作，而且在内部状态空间进行操作（一条已经被网络处理过的轨迹（a trace of what already been processed by the network，往下看就理解什么意思）），从这个意义上，回归神经元网络与前馈网络结构有本质的不同。这等价于一个迭代函数系统(IFS; see (Barnsley, 1993) for a general introduction to IFSs;  
(Kolen, 1994) for a neural network perspective)或者一个动力学系统(DS; see e.g. (Devaney, 1989) for a general introduction to dynamical systems; (Tino et al., 1998; Casey, 1996) for neural network perspectives)。状态空间让对于在未指定的（可能是无限的）时间间隔上的暂时的/循序的扩展相关性的表示（和学习）成为可能，依据



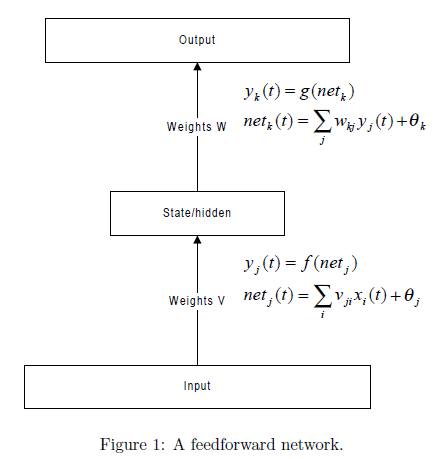
为了限制这篇文章的讨论范围，简化数学推导，我们假设网络按照离散的时间运行（完全有可能采用连续的时间）。可以证明，如果我们进一步假设权重都是有理的而且采用连续的输出函数，网络能够表示任何的图灵机（再一次假设任何隐藏节点的个数都是可取的）。很重要的一点是，我们马上就会看到所有这些都可以被一个离散时间的RNN计算和处理。甚至有证据显示，如果采用实权重系数（即这个网络是完全模拟的）,我们可以得到超过图灵机的处理能力。

**2.一些基本定义**

为了简化记法我们我们会限制网络仅包含两层（一个“隐藏层”或“状态层”，一个输出层，不包括输入层）。每一层都有自己的下标变量：输出层k，隐藏层j（和h），输入层i。在一个前馈网络中，输入向量x是通过一个权重层V进行传播的，



其中n是输入的个数，theta\_j是偏置，f是输出函数（任意可微的函数类型）。一个网络如图1所示。



在一个简单的递归网络中，输入向量也是相似地通过一个权重层传播，但是同时通过一个加性的递归矩阵U结合了先前的激活状态

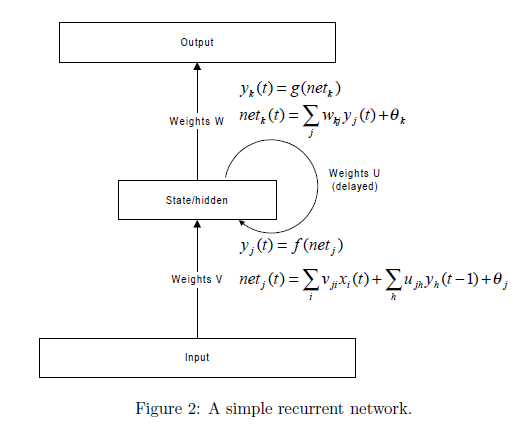


其中m是“状态节点”的数量。

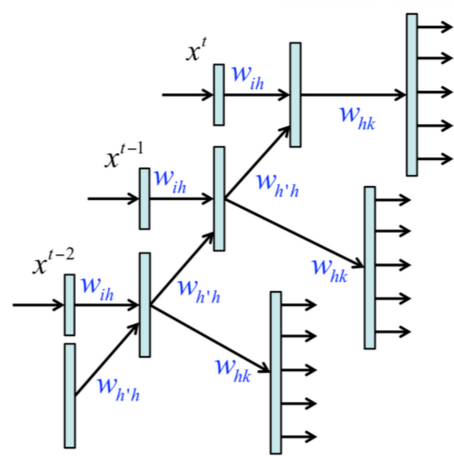
网络的输出是由状态和一系列输出权重W决定的



其中g是输出函数（很可能与f相同）。



一个更加直观的RNN表示，就是将RNN网络在时序上进行展开，如下图所示



**反向传播算法**

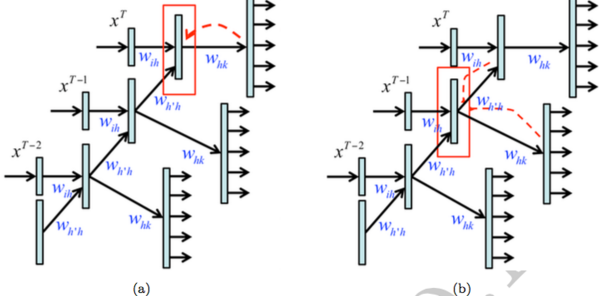
使用Backpropagation Through time (BPTT)算法对网络中各个参数求导，与传统的反向传播算法类似，只不过这里需要加入时序的信息。我们从t时刻逐层计算梯度。首先考虑从隐层到输出层的权重。计算损失函数对输出层的梯度的过程，则有



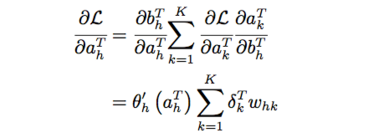
那么损失函数对的梯度为：



接着考虑。首先要计算损失函数对于各个隐层的梯度，这里分为两种情况，分别如下图a,b。



首先对情况（a），即t=T时刻。该时刻隐层的误差来子最后一个输出，则有：



对于情况（b），即t<T时，隐层的误差不仅来自输出层，还来自于下一时刻的隐层，因此有

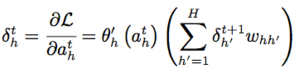


显然另=0，那么（a）和（b）两种情况可统一的写成一个式子，从而得到两个权重的梯度公式为：

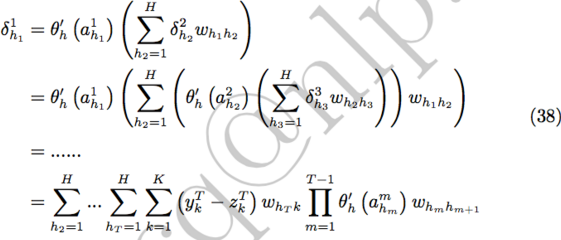


**RNN的踢的梯度消失/梯度爆炸问题**

通过上面的计算过程可以知道RNN中梯度的计算过程。我们再来看一下残差的计算过程：



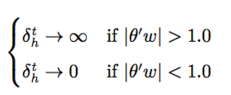
其中，我们根据残差的计算公式，推出t=1的残差，得到：



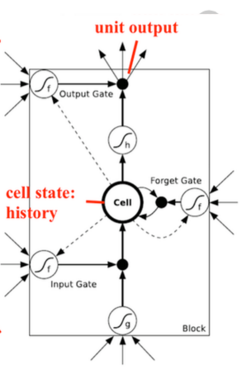
更一般的表示为



可以看到公式右边相当于有T个形式的连乘，那么当T越大时，会出现如下两种情况：



这也就导致了当RNN网络的时间序列很长时，会出现梯度爆炸或者梯度消失的情况。因此人们希望引入一种机制，保证。下面我们就将介绍一种新的RNN网络长短时记忆单元LSTM（Long Short Term Memory）。LSTM的结果如下图所示：



LSTM本质上也是一种循环神经网络，它在RNN的基础上，为隐层每个节点开了三扇门：

* Input gate:表示是否允许信息加入到当前隐层节点中，如果为1，则允许信息输入，否为不允许。从而起到屏蔽输入信息的作用；
* Forget gate：表示是否保留当前隐层节点存储的历史信息，如果为1则表示保留当前的信息，否则当存储的以前的历史信息清除
* Output gate:表示是否将当前节点输出值输出给下一层，如果为1，则表示将当前节点的输出值作用于下一层。

我们用如下符号表示三个门：



下面我们来看一下前向算法的过程：

用表示从节点i到节点j的权重。那么从cell state到三扇门的权重分别为。从书入门向量到三扇门的权重分别为：，从隐含层的输出到三扇门的权重分别为

用和分别表示t时刻节点j净输入和净激活。

用表示t时刻c个cell的state，也就是t+1时刻的历史信息

f表示三个Gate的激活函数，通畅情况下是Sigmoid函数。

用I,H和K分别表示输入层的节点、隐含层的cell个数和输出层的节点数。

则可以得到各个Gate前向传播函数：  
InputGate:



ForgetGate:



Cell:





OutpuGate:



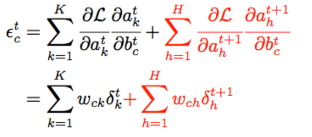
从而得到CellOutput为：



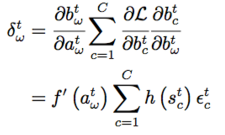
现在我们推导反向传播算法，首先来定义



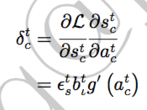
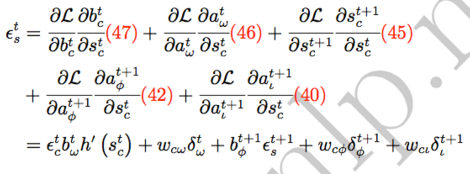
则cell Output为：



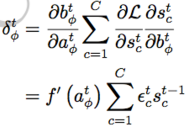
Output Gates:



Cells:



Forget Output:



Input Gate

