深度神经网络学习：AUTOENCODER之SPARSE CODING（稀疏编码）

[2015 年 5 月 7 日](http://blog.yinfupai.com/2174.html) [三符风云涌](http://blog.yinfupai.com/author/admin) [发表回复](http://blog.yinfupai.com/2174.html#respond)

　　这个是一个让人感觉很神异的技术，给一堆图片，然后用软件扫描，软件就能把图片中的特征找出来，这对人来说是很轻易的事，对软件来说，最大的问题就是，怎么应该进行才可以？

　　写软件的技术也不是多么高科技，最终需要用到的还是人的思想，然后把思想转化成代码，从而解决现实问题，编程可以说是一个验证思路的一个绝佳途径，在下甚至以为，新时期的道士，应该个个都掌握编程的技能。

　　说远了。

　　这个给了一堆图片，然后能提取出特征，应该怎么实现？

　　一般的问题想要智能解决，思路是这样的：给一组输入，再给一组输出，比如天气预报，输入的就是时间，输出就是天气，然后给神经网络任意赋上一堆值，再通过数学运算，看目标结果与需要的输出数据相差有多大，然后转化成微积分的方向，计算出运算路径中应该怎么反向调整数据才能拼凑出目标结果来，然后不断重复这个计算并再拼凑的过程，最后接近目标结果，得到满意答案而确定出来的神经网络中的参数，就可以理解为特性。

　　但是如果是一堆图片，这个怎么处理？如何提取图片中的特征，这里不得不说，有一个很有意思的解决方案。

　　它是这样进行的：

　　１、把图片切成各种小块，并把这些小块换成数据表达

　　２、用神经网络去运算这些数据

　　但问题来了，运算的数据的输出结果是什么？

　　这里就是极其聪慧的一点啊，输出的结果再进行一次神经网络运算，让它能输出为输入值。

　　输入－》运算（编码）－》运算（解码）－》输出

　　然后比较输入与输出，根据差值，去调整中间的编码运算与解码运算，而至于编码与解码到底在干什么是不用管的。

　　反正最后输入与输出结果精度达到满意要求时，所保留的就是特征了。

　　为什么会这样？其实原理很简单，中间的编码层的神经网络处理，相当于去将原图进行任意的干扰，而解码层的神经网络处理，相当于将干扰后的图像进行了恢复。

　　而这个过程有一个有意思的地方是，因为中间出现了干扰与解干扰的过程，输出等于了输入，多次变换后，保留下来能识别到原输入的就是特征。

　　同样的，可以在这种思路中，多加入一个混淆器，即随机添加大量的噪音，然后再消除掉，同样可以增强神经网络的搞干扰能力，而进行了干扰处理后的这种手法得到的编码器称为Denoising [AutoEncoder](http://blog.yinfupai.com/tag/autoencoder)s降噪自动编码器。

　　这真是一个令人惊叹的想法，而且只能在计算机上实现，传统的其它计算工具是无法实现这个的，因为它依赖大密度的运算。

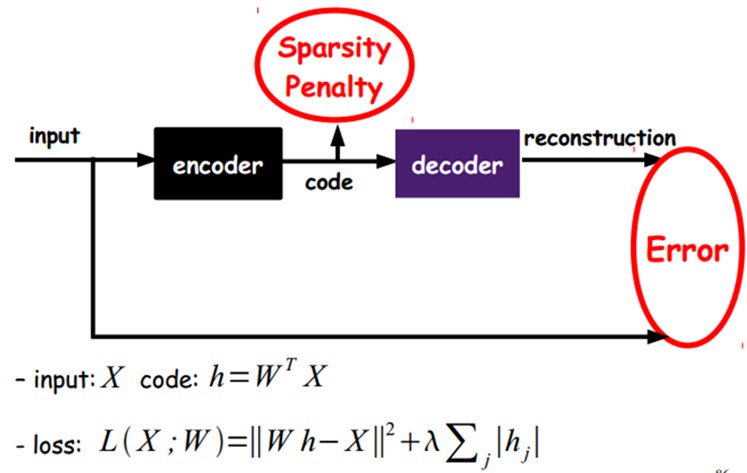
　　那么这样就够了么？

　　于是这里出现一种信念，这种特性的抽象是否可以尽量的简单再简单？

　　损之又损，以至于无为，无为而无所不为？

　　这个可能吗？

　　于是，如果坚持相信，再复杂的东西背后都有简单的规则的话，这里就可以基于这种信念再干一件事：**稀疏化**

[](http://blog.yinfupai.com/wp-content/uploads/1365439878_3585.jpg)

　　怎么做呢？因为复杂的背后总有简单的规则是不是？那么简单的规则是不是就意味着可以删减很多东西后，还只保留最核心的一些东西，就能复原绝大部分原来的信息？

　　既然是这样，那么就意味着，如果能找到一种，让神经网络里大部分神经元不起作用也能进行成功由输入得到输出结果时，那么就能得到这个。

　　所以，可以对神经元进行限制，每个神经元有一个激活值，然后规定所有的神经元的总激活值，不得超过一个数值，比如0.05，那么这样就会导致大部分神经元变成０值，而极少数神经元需要更加精准的计算出自己的权值。

　　那这个怎么实现？

　　先从源头说起，稀疏化，实际上就是寻找一组“超完备”基向量来更高效地表示样本数据。[稀疏编码](http://blog.yinfupai.com/tag/%e7%a8%80%e7%96%8f%e7%bc%96%e7%a0%81)算法的目的就是找到一组基向量 ，使得我们能将输入向量表示为这些基向量的线性组合。

　　首先，输入与输出必须要满足：Min |I – O|，其中I表示输入，O表示输出，即输入与输出相差小值。

　　如果用图片来说，就是能找到少量的、微小的图片来组合成这个大图，然后这堆微小的图片，称它为字典Φ，而任意抽一个小图片出来（比最微小的图片要大），那么这个小图片，就全塞进去，因为还可以比如是Φ1的20%与Φ2的50%与Φ3的50％共同构成了这个图(因为比最小图片大，所以加起来会超过100%)。

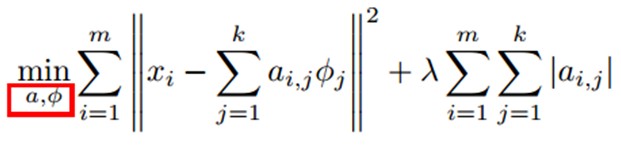
　　那么因为“要找到最小图片”所以就等于是加上了一个多的条件限制，因为O为输出，Ｏ就是那个抽出来的那个图需要求的目标图，而输入为Ｉ，所以就可以表达为：

　　I=x1+x2+x3+x3…..xi

      O = a1Φ1 + a2Φ2 + … +anΦn

　　然后考虑加入一个惩罚系数，这个用于调节网络里的值用。

　　那么实际上解决问题就是要把a和Φ拼凑出来，使得计算结果变成一个最小值。

[](http://blog.yinfupai.com/wp-content/uploads/1365483429_5706.jpg)

　　右边加号后的就是加入的稀疏代价函数，这个函数也可以用对数代价函数（利用柯西先验概率），即log(1+ai2)，它是用来对于远大于零的值进行限制用的，以避免出现网络中的值相差太大并集中，从而失去了稀疏化的目的。

　　因为目标最小值是稳定的，如果Φ或a中的值太大了的话，右边的代价函数就没什么意义了－－－会变得非常小。所以要对||Φ||2的值进行限制，让它小于某个值。

　　在实现中，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）。

      然后就这样反复计算就可以了。

[autoencoder理解(2): 自动编码器的作用之稀疏编码](http://blog.csdn.net/u011534057/article/details/53261920)

2016-11-21 18:01 1019人阅读 [评论](http://blog.csdn.net/u011534057/article/details/53261920#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/u011534057/article/details/53261920#report)

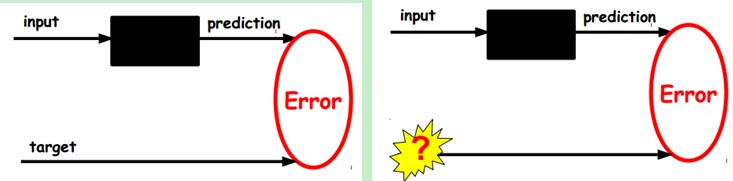
http://static.blog.csdn.net/images/category_icon.jpg 分类：

VAE（4） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

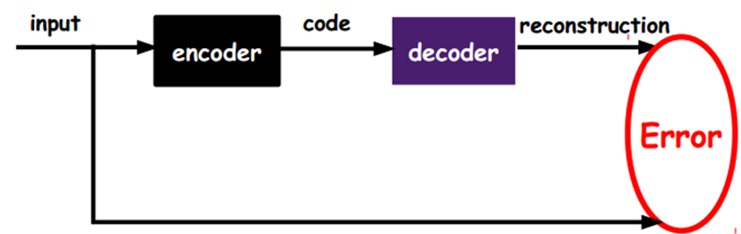
如果给定一个神经网络，我们假设其输出与输入是相同的，然后训练调整其参数，得到每一层中的权重。自然地，我们就得到了输入I的几种不同表示（每一层代表一种表示），这些表示就是特征。自动编码器就是一种尽可能复现输入信号的神经网络。为了实现这种复现，自动编码器就必须捕捉可以代表输入数据的最重要的因素，就像PCA那样，找到可以代表原信息的主要成分。

具体过程简单的说明如下：

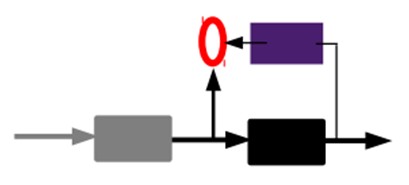
1）给定无标签数据，用非监督学习学习特征：



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？

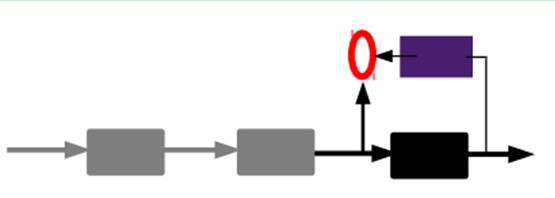


   如上图，我们将input输入一个encoder编码器，就会得到一个code，这个code也就是输入的一个表示，那么我们怎么知道这个code表示的就是input呢？我们加一个decoder解码器，这时候decoder就会输出一个信息，那么如果输出的这个信息和一开始的输入信号input是很像的（理想情况下就是一样的），那很明显，我们就有理由相信这个code是靠谱的。所以，我们就通过调整encoder和decoder的参数，使得重构误差最小，这时候我们就得到了输入input信号的第一个表示了，也就是编码code了。因为是无标签数据，所以误差的来源就是直接重构后与原输入相比得到。



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）。

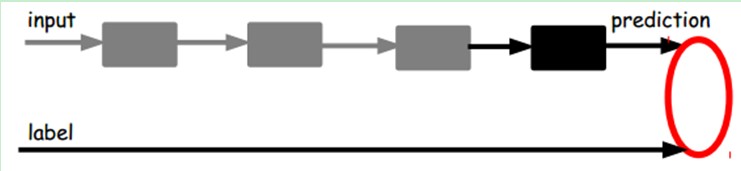


3）有监督微调：

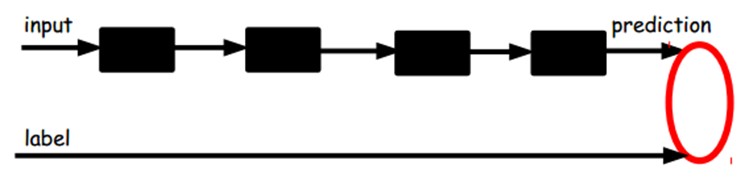
经过上面的方法，我们就可以得到很多层了。至于需要多少层（或者深度需要多少，这个目前本身就没有一个科学的评价方法）需要自己试验调了。每一层都会得到原始输入的不同的表达。当然了，我们觉得它是越抽象越好了，就像人的视觉系统一样。

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）



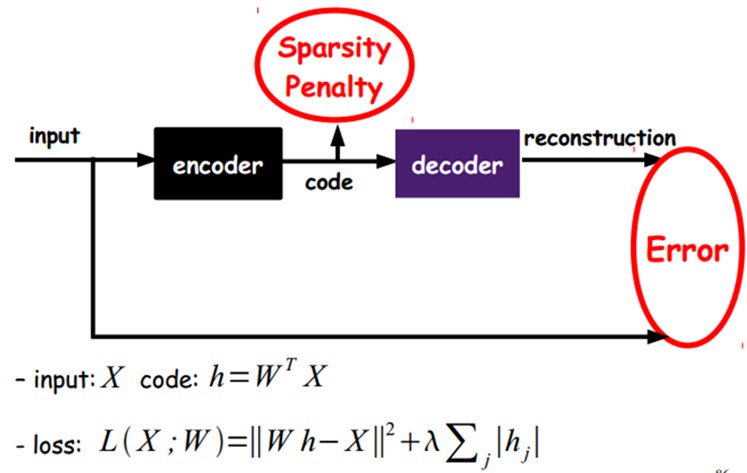
一旦监督训练完成，这个网络就可以用来分类了。神经网络的最顶层可以作为一个线性分类器，然后我们可以用一个更好性能的分类器去取代它。

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**](http://lib.csdn.net/base/datastructure)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。

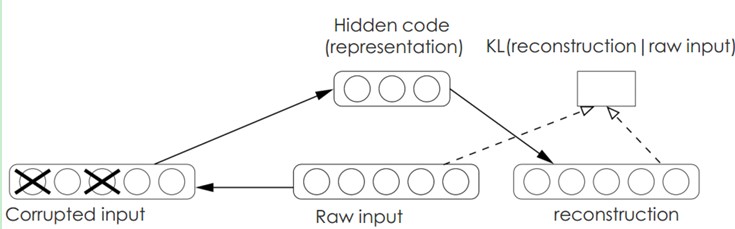


上面公式中：h是编码参数

 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。

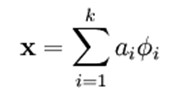


9.2、Sparse Coding稀疏编码

  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。



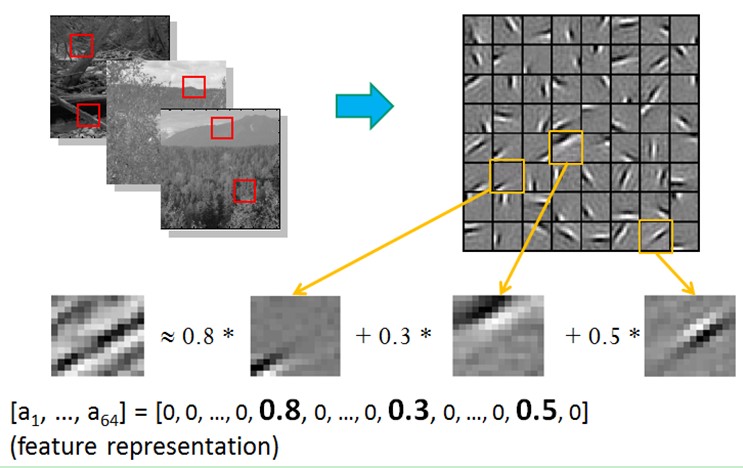
因此，它们可以用来表达输入I，这个过程也是自动学习得到的。如果我们在上述式子上加上L1的Regularity限制，得到：

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0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0.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。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。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。



比如在图像的Feature Extraction的最底层要做Edge Detection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（稀疏矩阵）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”。

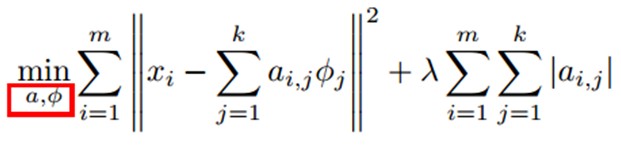
  这里可能大家会有疑问，为什么把底层作为Edge Detector呢？上层又是什么呢？这里做个简单解释大家就会明白，之所以是Edge Detector是因为不同方向的Edge就能够描述出整幅图像，所以不同方向的Edge自然就是图像的basis了……而上一层的basis组合的结果，上上层又是上一层的组合basis……（就是上面第四部分的时候咱们说的那样）

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。）

  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。



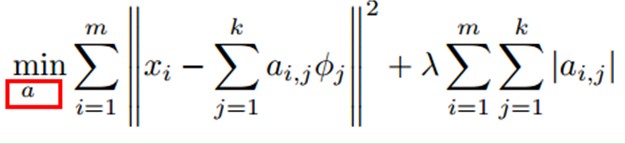
每次迭代分两步：

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：

