CNN

2015年10月5日 星期一　　下午7:27

一、简介

CNN（convlution nerual network，卷积神经网络）是一种应用于图像、语言、NLP等领域的神经网络，在图像中的用法比较经典。

他的特点在有：

A 某些层与下层的神经元之间的连接不是全连通，而是一种特殊的网络连接方式，这些层叫“卷积层”(convlution layer)

B 卷积层与下层神经元的连接中，某些连接(可以看做是参数/权值)是共享的(相同的)

C 因为有这样特殊的设计，可以从特征提取的效果、降低参数规模、防止过拟合等角度提升网络效果

CNN主要精髓在于convlution layer，其他一些层、一些变换等，也起到了重要的辅助作用。

二、历史来历

1、感受野、可视域（详见参考资料LeNet）

机器学习的人工神经网络就是受人脑神经网络的影响，CNN也借鉴了神经学方面的概念。卷积网络最初是受视觉神经机制的启发而设计的，是为识别二维形状而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。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。

三、convlution layer

1、全连通隐层的分析

A 全连通作用

从一般的全连通层出发（需要预先看看autoencoder的特征压缩/提取等功能），在这里，全连通层可以看做是"一种对下面一层输入信号(比如，输入层)做特征组合/特征压缩/特征抽取的过程" 等，但是这样的隐层是全连通的，每个下层神经元和本层神经元之间都有一个连接。

B 为什么全连通

为什么要"全连通"呢？为什么要两两都做连接呢？其实这是一种偷懒但是保险的做法。因为我们做的是提取特征的过程，这个特征权重我们是要学出来的，学之前我们不知道哪些特征的信息是有用的，他们都有可能是有用的，

系性我们就把所有的节点之间都用参数连接起来，如果他真的没用，学出来的权重会非常低，这对我们是不影响的。

C 全连通的一个可优化点

这样看来，如此设计没问题，但是我们还有很多优化空间。

其实，在我们对输入没有任何先验知识的情况下，我们不知道哪些神经元之间的信息是有用的，哪些是没用的。但是我们并不是一无所知的，我们还是有些先验，我们可以把这些先验，想办法加到网络结构中，这个其实就是使用卷积的初衷，下面会细讲。

D 全连通弊端

如果我们想增加一些信息（让layer的input窗口变大）、多提取一些特征（让layer的output维度变大），势必会增加全连通的连接数，也就是增加了参数规模，在image中经常有81*81级别的input，81*81*emb_dim*hidden_dim个参数，这规模太庞大，而且容易过拟合，学出的特征权重可能过拟合输入特征。

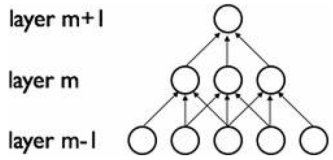
2、用卷积层代替全连接层

为了避免这些，我们考虑全连通是否可以简化。

有一些先验知识，我们可以利用。我们发现一些局部空间内，特征的组合方式/提取方式是类似的（尤其是在image中，比如一个局部视野内构图有时候是相似的，图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在另一个区域同样适用）。

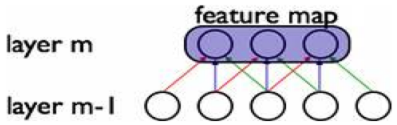
所以我们考虑共享这个特征组合/特征提取的权重参数，这也就是全连通层 -> 卷积层的操作。

替换的思路包括两种思想（一起作用）：



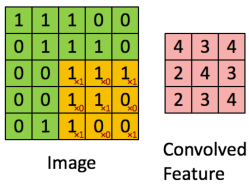
A、稀疏连接代替全连接（sparse connectivity）

不用全连接，只用一部分参数连接。如右边两幅图



B、权值共享（shared weights）

即相同位置的稀疏连接用同一个参数做权重


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
Image
4 3 4
2 4 3
2 3 4
Convolved
Feature

即相同位置的稀疏连接用同一个参数做权重。

网络中前馈的过程中(提取特征的过程)是这样的，我们之前的全连接隐层是一个固定的特征权重的参数矩阵，而CNN的隐层是特征权重的参数矩阵，可以看做一个滑动窗口。在输入层网络位置固定的情况下，滑动窗口会对输入层网络遍历一遍，例如右上图中绿色的是输入层网络节点，黄色是滑动窗口覆盖到的节点，覆盖到某一批节点的时候，会以这些节点为输入做一次参数提取，提取的结果存下来，下一节的pooling部分会用到。
在权重共享的条件下如何运行和计算的过程，详见参考资料1中的一个gif动画。

由于滑动窗口中，相对位置相同的节点，所使用的参数是一样的，这就是权值共享。

这部分权重覆盖一个窗口，相当于是把每个小的窗口进行特征提取，得到一个值（得到的是值这里体现了稀疏性）。这个小的窗口是一个可视域，这里借鉴了可视域、感受野的概念。

3、补充说明

注意，这里相当于卷积减少了特征组合的多样性，如果按照上面讲的这种模式，只有一种特征组合，特征组合这方面看似是减少了，但后续会引入feature-map解决这问题

把每个可视域提取特征的结果拿过来看，看输入、输出(执行过pooling之后)的神经元个数、隐层维度没有太大改变(这样保证了特征表示能力没有太大减弱)，但是层与层之间的参数规模减小了很多，这就是卷积的一大贡献。

四、pooling

1、卷积之后的输出

现在我们得到的卷积层的节点是，每滑动一个窗口之后，做一个特征提取，会有一个输出。最终的输出节点为：滑动窗口覆盖的可视域数 (也就是窗口滑动的次数+1)，这个规模也是很庞大的。我们可以想办法降低这个规模。

2、降低输出的规模

其实各个特征提取的规则都一样，说明这些特征值有一定的可比性。
对待这些输入不一样，特征提取过程一样的，在这个卷积的阶段其实我们想得到的是整个输入的表示（整个图片的含义/整句话的意思）各个局部的特征还是需要进行融合的，一个很自然的想法就是可以对各个位置提取出的特征进行聚合一下，这个聚合因为特征之间是可比的。

3、降低输出的方式

通常使用mean-pooling/max-pooling（平均/最大），就是各个可视域的提取特征的值简单的取mean/max

这种mean/max的概要统计，有两个好处：

A 降维（多个可视域的东西聚合在一起）

B 防止过拟合

五、drop-out

drop-out是一种防止过拟合的策略。（博客/教程没有说，详见维基）

1、训练

会随机的抛弃一些节点，以一定的概率P保留节点(以1-P的概率抛弃节点)，抛弃节点不参与本次的计算，这个节点就是layer上的nodes，同时nodes的输入/输出权重，这次也不参与计算了。

对隐层做drop-out的时候，节点抛弃一般是50%的概率，可能会有一半的节点被抛弃；

对输入层(embedding层)做drop-out的时候，节点抛弃的会少一些，P>50%(P是保留的概率)，因为如果他们抛弃了就是信息直接没有了。

2、预测

预测的时候也有对应的策略，由于训练的过程是不确定的，n个节点都有可能被抛弃，对应有2^n种抛弃的可能。如果我们在预测的时候，遍历这2^n种，然后做模型combine，复杂度非常大。

这里采用的策略是，用对dropout的估计代替采样，换取每次预测都是一个确定性的预测；

就是每个nodes都乘以保留的概率P，这样的到的是，对采样过程的一个估计。

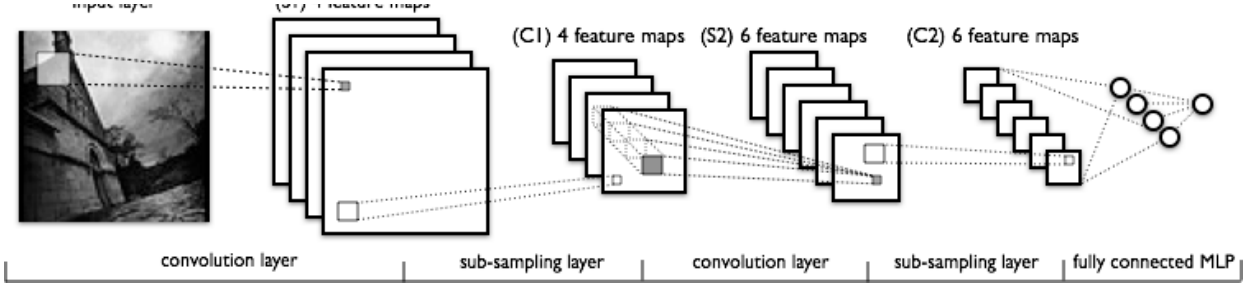
dropout避免了所有的nodes见过所有的数据，这样的话可以避免过拟合，而且可以提高训练速度。

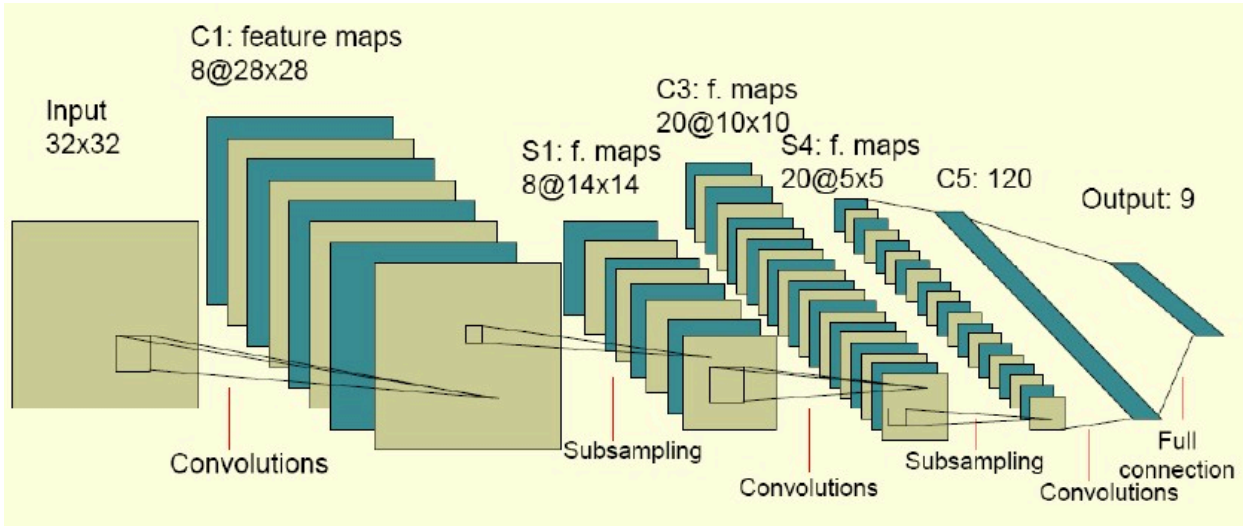
六、feature map

由于卷积+pooling只是解决把一系列的输入，用一种特征提取的方式进行提取，得到一个输出的值；所以这种方式就有个问题:特征提取方式单一，这样很多信息捕捉不到，进而我们提出featuremap。

feature-map中包含了很多个卷积+pooling的操作，这些特征进行组合，可以得到不同的提取特征的方式。

七、LeNet的例子





比较经典的一个图片是上面图，彩色的那副中，下面分析一下上图：

Input：首先是对input的图片（32*32像素的）（在NLP中应该是一个长度为32*32=1024的句子）进行处理，遍历整个句子，每到一处取一定的窗长（这里是5*5的（32-28+1）），这个窗口也可以看做可视域/接受域。

convlution：这个窗长（5*5））会经过卷积的过程，投射到下一层上，每个窗口对应下一层的一个点（一共28*28个窗口），注意这里featuremap中有8个feature，相当于做了8变上述的卷积过程（每次卷积对应的参数(权重)不一样）。

subsampling：相当于从28*28->14*14的一个映射，映射的过程中是一个子抽样和局部平均（mean-pooling？），一个2*2->1*1的过程，这不仅仅是映射/求平均，还包括一个可训练的参数/权重、一个sigmoid、一个可训练的bias，这个过程的可视域是2*2.

convlution：又是一个卷积，而且窗长是5*5，这个是14*14->10*10，卷积过程同上；值得注意的是，featuremap中feature的个数8->20，这也就是说每个前一层的神经元可能和后一层的多个神经元相连（通过不同的feature的权重，连到不同的feature上，但具体怎么连，没说）

subsampling：和上述的过程一样，不细讲
Convlution:好像是一个卷积的过程，它由 120 个神经元组成，每个神经

元指定一个 5×5 的接受域，好像是，但是没理解

全连通：从120->9是一个全连通的矩阵，让这两层全连通，最终是一个维度为9的output。

图中所示的多层感知器包含近似 100000 个突触连接，但只有大约2600个自由参数。VC维显著降低，这样他的泛化能力更强。

八、相关论文

1、是第一篇专门用CNN做NLP的论文，oxford 做情感分类

http://nal.co/papers/Kalchbrenner_DCNN_ACL14

2、是在NLP领域第一次使用CNN的论文，对CNN的介绍比较少

标题： Natural Language Processing (almost) from Scratch

九、参考资料：

1、（推荐）UFLDL的卷积，里面有提到auto encoder，详见另一篇笔记

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

2、UFLDL的pooling 推荐 尤其maxpooling讲的不错

http://ufldl.stanford.edu/wiki/index.php/Pooling

3、（推荐）中文博客（基本按照UTLDL的思路，加一些LeNet的东西，适合没事看看回顾，也适合对比自己理解怎么样）

http://blog.csdn.net/stdcoutzyx/article/details/41596663

4、LeNet: convolution一般 max-pooling讲不好 多核卷积那幅图适合好好看看（包含feature-map的）

http://deeplearning.net/tutorial/lenet.html

5、中文博客（和LeNet比较接近），开头的原理介绍和LeNet细节介绍很赞！

http://ibillxia.github.io/blog/2013/04/06/Convolutional-Neural-Networks/

6、wiki:

https://en.wikipedia.org/wiki/Convolutional_neural_network

7、其他一些

http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf