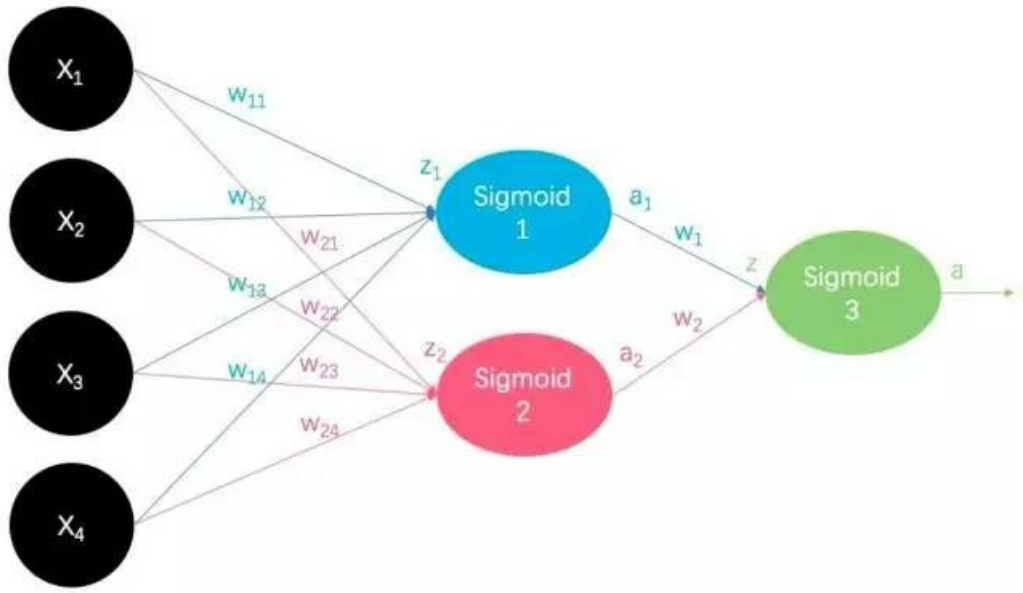
不要再纠结卷积的公式啦！0公式深度解析全连接前馈网络与卷积神经网络

原创 夕小瑶 夕小瑶的卖萌屋 2017-08-08

Hello~你们的小夕终于吐泡泡了～前几天小夕又加班赶project啦，拖了好几天，都快患上拖稿焦虑症了_(:з」∠)_

关于卷积神经网，小夕就不从卷积讲啦。以前不止一个粉丝问我卷积神经网络中卷积的意义，甚至在知乎上被邀请回答一个“卷积神经网络为什么不叫互相关神经网络”的类似的奇怪问题，终于忍不住，还是赶紧把CNN写了吧（说的自己要去拯救世界了一样\(//∇//)\

我们还是从前面更简单的机器学习model开始。回顾一下前面已经讲过N*N*N次的全连接前馈神经网络，前面文章中小夕讲过，对于一个有一个隐含层的全连接前馈网络：



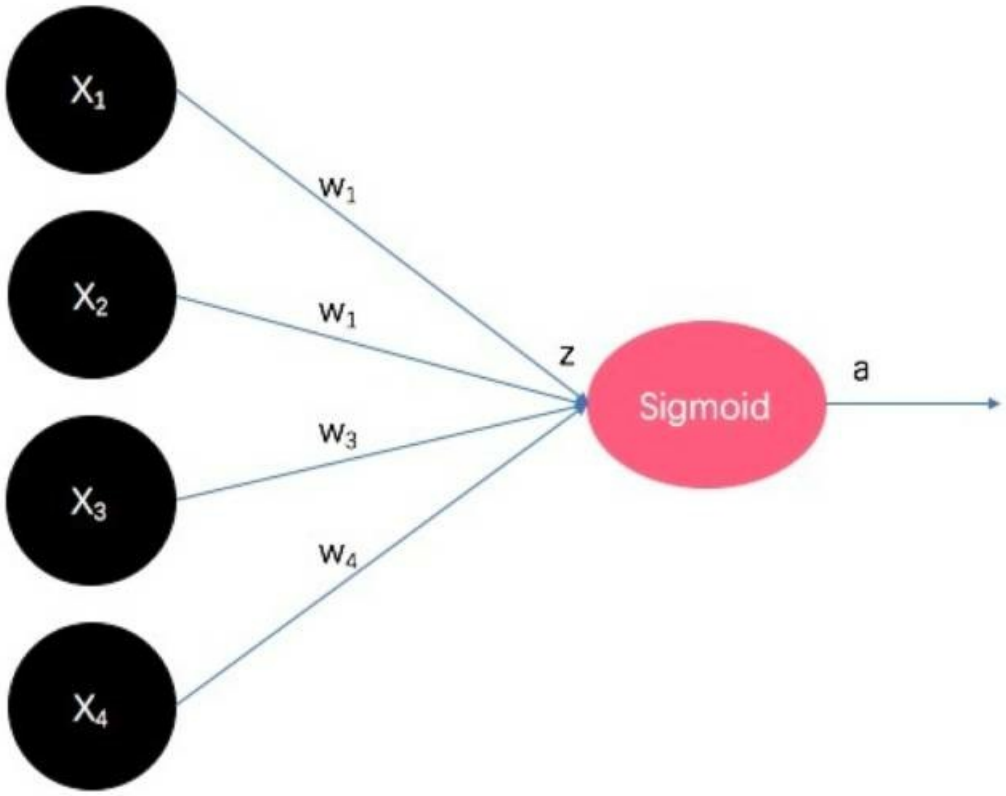
这里就可以看作是两层简单的分类器的前后级联，前一层分类器的输出就是后一层分类器的输入，那么显然前一层分类器的每个输出（即每个隐单元）代表什么含义我们是不清楚的，也就是前一层分类器学到的是分类未知意义的类别！而后一层分类器则直接利用前一个分类器得到的未知类别来学习输出最终我们定义的类别！举个栗子。

比如输入是一副图像：

这个图像假设是100*100的，也就是有10000个像素点。每个像素点取值0-255。

试想一下，如果我们不想人为定义特征，想要直接将原始图像丢进去，去分类图像是否包含狗这个类别。那么这时就相当于输入层有10000维，也就是有10000个特征，每个特征就是一个像素点的值。

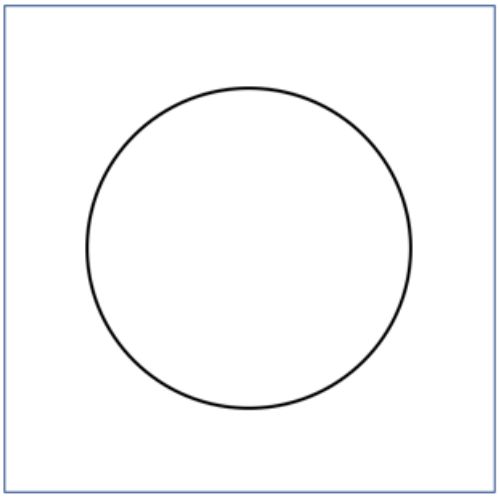
如果我们的机器学习模型不加隐含层的话：



这时的模型显然是将每个像素点直接连接到了“是不是狗”这两个最终类别上。然而我们稍微想一下也知道，其实每个像素点的值跟是不是狗并没有什么联系（你不能说这个像素点是黑的（值为0），就说这个像素点是狗身上的，同样，像素点是白的（值为255）也不能说明这个像素点是不是狗身上的。）所以显然直接用每个像素点的值做特征去决策是不是狗这件事是非常不靠谱的！（每个特征都跟类别关系不大啊姐姐）

但是，如果我们加一个隐含层呢？这样情况会不会好一些呢？

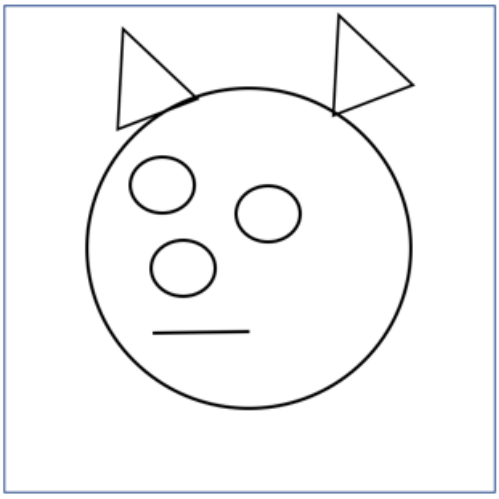
设想一下，如前所述，加一个隐含层后，模型可以学习出一层未知类别，而这些类别完全可以做到跟像素点强相关！比如一个隐含类别是“以图像中心为圆心，半径为50的地方是否有一个圆形”



学习出这个分类器很简单，model只需要让这个圆圈所在的像素点对应的权重很大（比如2），让其他特征的权重接

近0。这样这个地方出现的圆越明显（像素值越接近0），就会导致该子分类器的输出接近0，而这个地方没有出现圆的时候（像素值接近255）就会导致分类器的输出很大，看，很轻松的就学会了这个简单的分类任务吧（当然，为了学到这个地方有没有圆，还需要考虑圆周围的像素点（要有足够对比度才能说明这里真有圆哦），不过忽略这些细节啦，懂了小夕瑶表达的意思就好）。

好啦，这个子分类器训练好啦，也就是一个隐节点弄好了，那么同样的道理，其他的子分类器（隐节点）也可以学习到一些奇怪而简单的隐含类别，这一系列的类别组合起来完全有可能是这样子的



看，基于这些类别（也就是对下一级分类器而言的特征），下一级分类器就很容易分类这个图片是不是狗啦～比如上面这个图片中，就大约有7个隐节点，分别负责7个线条存在与否的分类决策。那么下一级分类器只需要让这7个特征的权重都大一些，这样当这些特征全部存在的时候，显然这就是一条狗啊！那么后一级分类器就能很自信的决策说：“这图片是一条狗！”看～这比直接让单个像素点与狗挂钩的做法科学多了，自信多了吧～

这就是用深度全连接前馈神经网络做分类的基本原理。

但是！你一定能发现问题！这样显然有很大的局限性！比如狗一旦换个姿势呢？狗一旦换个大小呢？图像中的狗蜷缩在一个角落呢？

显然！这时全连接前馈网络的隐含层的任务量要爆炸了！要有非常多非常多的隐节点来学到非常多隐含类别/隐含特征才可能足够应付这么多复杂局面！

而隐含层节点大量增多后将会导致神经网络的参数迅速增加！比如上面这个例子，增加一个隐节点就要增加10000+2个参数，显然代价是非常大的。那么有没有更好的解决方案呢？

很显然啊！既然我们的简单分类器要学的是一个简单的圆，一个简单的直线，一个简单的决策任务，那么所有的学习圆的隐节点完全可以合并为一个节点啊！这时我们可以用一个远小于整幅图像的“窗”来表示。比如就用一个20*20的窗（这样就只有400个参数，而之前全连接的时候要100*100=10000个参数），而且这个窗就负责找出图片中各处的小圆！这个“窗”就叫“**卷积核**”，（显然本质上就是一个缩小版的输入到一个隐节点的连接权重）为了寻找图像中各个角落的小圆，我们就可以让这个卷积核依次滑过图像的各个角落，只要在某处发现了小圆，就在该处激活，即标记好这里有个小圆。

因此，就跟前面全连接的时候一样，**为了学到多个特征，我们肯定要设置多个卷积核呀**～每个卷机核负责一种简单

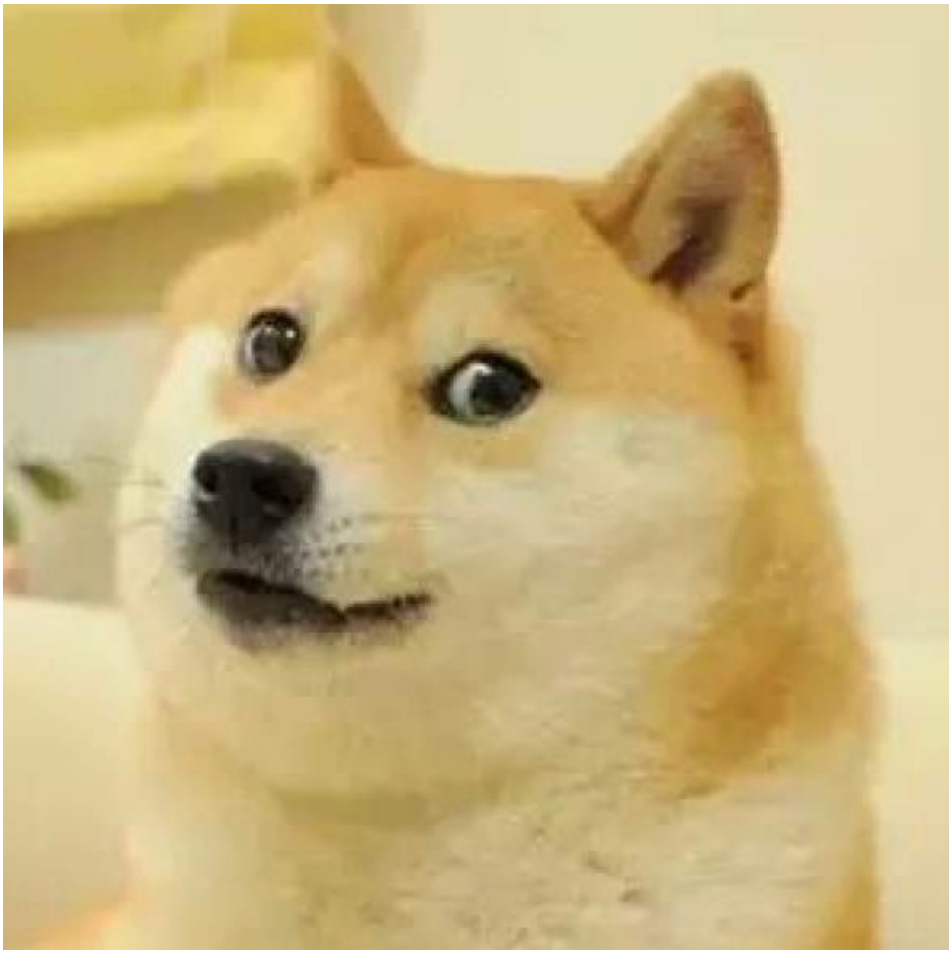
的分类任务（说到这个份上了，大家也能猜到，这依然是跟全连接的时候一样，这里的简单分类任务其实就是为下一层的子分类器抛弃旧特征，创造新特征啦）。

显然，依然跟全连接时一样，比如同样是20*20的卷积核，不同的参数就代表提取出了不同的特征，有的负责提取20*20块里的小圆，有的负责提取三角，有的提取直线等等～同一组大小的多个卷积核，就统称为一个20*20的**滤波器(filter)**（即一个某size的滤波器下可以设置学习多个卷积核）

而我们既然需要从20*20的块里去分类小圆，因此当然也可能需要从50*50的块里寻找大圆啦～因此**在一个卷积层，我们还可以设置多个size的滤波器**～当然每个size的滤波器下可以设置多个卷积核来提取不同特征。

我们再来考虑更复杂的情况！

我们知道，很多时候的输入并不是只有一个表示层！比如彩色图像就会包含红色、蓝色、绿色这三个图层，而不像前面的灰度图一样仅仅包含一个图层。



有时候彩色图像的一个圆圈仅仅出现在蓝色图层里，而没有出现在其他两个图层，因此显然如果我们的负责提取圆圈的卷积核仅在一个图层里滑动的话，可能会在很多位置遗漏很多信息，因此当输入的数据可以表示成很多层（即有很多不同角度的数据表示）的时候，**卷积核要在每个位置处把所有的图层都映射/"卷积"一下，并且求和**，才是真正确定图像在这个位置处到底有没有该卷积核要找的特征的做法。这里输入的多个层被称为多个**输入通道(channel-in)**，**所以一个卷积层，不仅可以设置多个size的滤波器，还可以在每个size的滤波器下设置多个卷积核，还可以让同一个卷积核每次同时考虑输入数据的多个通道！**

我们再来考虑更更复杂的情况！

假如我们现在的分类任务变了！变得更难了！现在我们直接想识别出一个图片中是不是正在发生**猫狗大战**！

这样的话！我们要有好多好多分类器啊！我们要识别猫，要识别狗，要识别头上的包，要识别创可贴等等，这么多子分类任务怎么办呢？能不能直接纳入一个卷积层呢？

当然可以啦！既然都可以有多个输入通道，当然也可以设置多个**输出通道（channel-out）**啊！一个输出通道就代表着一个子分类任务～（当然啦，每个子分类任务都会有它的一套filter及其一堆卷积核）。（当然啦，这些子分类任务也是人类不清楚的，神经网络自己知道）

至此，完整的卷积层就定义完成啦。我们再总结一下，一个卷积核在每个位置处要同时考虑所有的channel-in；然后一个size的filter下可以设置多个卷积核用于提取不同特征；然后可以设置不同size的多个filter来控制特征提取的粒度；然后可以设置多个输出通道channel-out表示多个分类任务。而且跟全连接前馈网络一样，被卷积核映射完（即线性映射结束后）别忘了丢激活函数哦～

所以在图像中，一个卷积层就分成多个size的filter，每个size的filter就对应着一个channel-in*width*height*channel-out的4D参数Tensor，其中的width和height就是卷积核这个窗口的宽和高啦～这种二维窗口的卷积核也称为2D卷积，同理三维卷积窗口就是3D卷积，参数Tensor就是5D的啦。

想一想，还有没有什么需要解决的问题呢？

刚才我们讨论的对象仅仅是针对卷机核在一个位置上的操作！显然一个卷积核滑过整个输入数据后，会在各种产生很多输出，那么这么多输出怎么取舍呢？

试想一下，其实我们要找出来猫的话，不管猫的位置是在图片左上角的角落，还是右下角，还是铺满整个图片，其实我们都说这个图片中包含了猫！大部分时候我们对它的位置不感兴趣，仅仅是关心这个图片中到底有没有这个东西，因此我们只需要判断该卷积核在所有的位置点产生的最强输出有多强就够啦～而其他位置点的输出直接抛弃就好咯～这个操作就叫**最大池化（max-pooling）**！这也是为什么大部分情况下最大池化是最有效的池化方式。

显然，处理一个卷积核的所有位置点，除了取最大值的方式，肯定还有一些场景需要其他更合理的方式。这些方式都叫**池化**。

除了最大池化，还有时会用均值池化（average-pooling）、取前n个最大值池化（max-n　pooling）等，从名字就能猜到池化方式啦，就不一一啰嗦了～当然，对所有位置点进行一次池化的话叫全局池化，相当于提取了全局的一个

特征或者说一个类别。如果我们对池化操作也定义一个范围（即一个窗/一个池化核）的话，就是局部池化啦～得到的就是局部特征/类别。

哦对了，都已经讲解到这里啦，很显然池化是肯定跟在卷积层后面的（当然，由于卷积后务必跟一个激活函数保证模型的非线性，所以如果激活函数也算一层的话，池化层是跟在激活层后面的）。

好啦～卷积神经网络讲完了，卷积-激活-池化，就这么简单。当然，正如之前所说，如果池化的结果仍然是作为一个隐含类别的话（即池化的输出不接我们最终的分类任务的话），那么这依然跟前面一样，可以作为下一层的特征去使用。因此池化层后面当然又可以接新一轮的卷积-激活-池化，也就是形成真正意义上的深度神经网络。