**1 前言**

    2012年我在北京组织过8期machine learning读书会，那时“机器学习”非常火，很多人都对其抱有巨大的热情。当我2013年再次来到北京时，有一个词似乎比“机器学习”更火，那就是“深度学习”。

    本博客内写过一些机器学习相关的文章，但上一篇技术文章“LDA主题模型”还是写于2014年11月份，毕竟自2015年开始创业做在线教育后，太多的杂事、琐碎事，让我一直想再写点技术性文章但每每恨时间抽不开。然由于公司在不断开机器学习、深度学习等相关的在线课程，[耳濡目染](https://www.baidu.com/s?wd=%E8%80%B3%E6%BF%A1%E7%9B%AE%E6%9F%93&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)中，总会顺带着学习学习。

    我虽不参与讲任何课程（我所在公司“[七月在线](https://www.julyedu.com/)”的所有在线课程都是由目前讲师团队的17位讲师讲），但依然可以用最最小白的方式 把一些初看复杂的东西[抽丝剥茧](https://www.baidu.com/s?wd=%E6%8A%BD%E4%B8%9D%E5%89%A5%E8%8C%A7&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)的通俗写出来。这算重写技术博客的价值所在。

    在dl中，有一个很重要的概念，就是卷积神经网络CNN，基本是入门dl必须搞懂的东西。本文基本根据斯坦福的机器学习公开课、cs231n、与七月在线寒老师讲的5月dl班第4次课[CNN与常用框架视频](https://www.julyedu.com/video/play/42/206" \t "_blank)所写，是一篇课程笔记。

    一开始本文只是想重点讲下CNN中的卷积操作具体是怎么计算怎么操作的，但后面不断补充，包括增加不少自己的理解，故写成了关于卷积神经网络的通俗导论性的文章。有何问题，欢迎不吝指正。

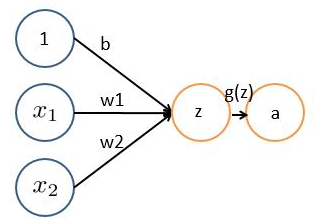
**2 人工神经网络**

**2.1 神经元**

    神经网络由大量的神经元相互连接而成。每个神经元接受线性组合的输入后，最开始只是简单的线性加权，后来给每个神经元加上了非线性的激活函数，从而进行非线性变换后输出。每两个神经元之间的连接代表加权值，称之为权重（weight）。不同的权重和激活函数，则会导致神经网络不同的输出。

    举个手写识别的例子，给定一个未知数字，让神经网络识别是什么数字。此时的神经网络的输入由一组被输入图像的像素所激活的输入神经元所定义。在通过非线性激活函数进行非线性变换后，神经元被激活然后被传递到其他神经元。重复这一过程，直到最后一个输出神经元被激活。从而识别当前数字是什么字。

    神经网络的每个神经元如下



    基本wx + b的形式，其中

* https://img-blog.csdn.net/20160720151554838、https://img-blog.csdn.net/20160720151607869表示输入向量
* https://img-blog.csdn.net/20160720151620525、https://img-blog.csdn.net/20160720151633098为权重，几个输入则意味着有几个权重，即每个输入都被赋予一个权重
* b为偏置bias
* g(z) 为激活函数
* a 为输出

    如果只是上面这样一说，估计以前没接触过的十有八九又必定迷糊了。事实上，上述简单模型可以追溯到20世纪50/60年代的感知器，可以把感知器理解为一个根据不同因素、以及各个因素的重要性程度而做决策的模型。

    举个例子，这周末北京有一草莓音乐节，那去不去呢？决定你是否去有二个因素，这二个因素可以对应二个输入，分别用x1、x2表示。此外，这二个因素对做决策的影响程度不一样，各自的影响程度用权重w1、w2表示。一般来说，音乐节的演唱嘉宾会非常影响你去不去，唱得好的前提下 即便没人陪同都可忍受，但如果唱得不好还不如你上台唱呢。所以，我们可以如下表示：

* https://img-blog.csdn.net/20160720151554838：是否有喜欢的演唱嘉宾。https://img-blog.csdn.net/20160720151554838 = 1 你喜欢这些嘉宾，https://img-blog.csdn.net/20160720151554838 = 0 你不喜欢这些嘉宾。嘉宾因素的权重https://img-blog.csdn.net/20160720151620525 = 7
* https://img-blog.csdn.net/20160720151607869：是否有人陪你同去。https://img-blog.csdn.net/20160720151607869 = 1 有人陪你同去，https://img-blog.csdn.net/20160720151607869 = 0 没人陪你同去。是否有人陪同的权重https://img-blog.csdn.net/20160720151633098 = 3。

    这样，咱们的决策模型便建立起来了：g(z) = g( https://img-blog.csdn.net/20160720151620525\*https://img-blog.csdn.net/20160720151554838 + https://img-blog.csdn.net/20160720151633098\*https://img-blog.csdn.net/20160720151607869 + b )，g表示激活函数，这里的b可以理解成 为更好达到目标而做调整的偏置项。

    一开始为了简单，人们把激活函数定义成一个线性函数，即对于结果做一个线性变化，比如一个简单的线性激活函数是g(z) = z，输出都是输入的线性变换。后来实际应用中发现，线性激活函数太过局限，于是人们引入了非线性激活函数。

**2.2 激活函数**

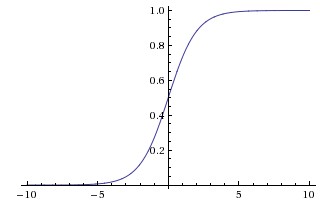
    常用的非线性激活函数有sigmoid、tanh、relu等等，前两者sigmoid/tanh比较常见于全链接层，后者relu常见于卷积层。这里先简要介绍下最基础的sigmoid函数（btw，在本博客中SVM那篇文章开头有提过）。

    sigmoid的函数表达式如下

https://img-blog.csdn.net/20160703105637734

    其中z是一个线性组合，比如z可以等于：b + https://img-blog.csdn.net/20160720151620525\*https://img-blog.csdn.net/20160720151554838 + https://img-blog.csdn.net/20160720151633098\*https://img-blog.csdn.net/20160720151607869。通过代入很大的正数或很小的负数到g(z)函数中可知，其结果趋近于0或1。

    因此，sigmoid函数g(z)的图形表示如下（ 横轴表示定义域z，纵轴表示值域g(z) ）：

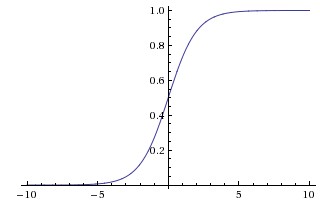


    也就是说，**sigmoid函数的功能是相当于把一个实数压缩至0到1之间。当z是非常大的正数时，g(z)会趋近于1，而z是非常大的负数时，则g(z)会趋近于0**。

    压缩至0到1有何用处呢？用处是这样一来便可以把激活函数看作一种“分类的概率”，比如激活函数的输出为0.9的话便可以解释为90%的概率为正样本。

    举个例子，如下图（图引自Stanford机器学习公开课）

    z = b + https://img-blog.csdn.net/20160720151620525\*https://img-blog.csdn.net/20160720151554838 + https://img-blog.csdn.net/20160720151633098\*https://img-blog.csdn.net/20160720151607869，其中b为偏置项 假定取-30，https://img-blog.csdn.net/20160720151620525、https://img-blog.csdn.net/20160720151633098都取为20

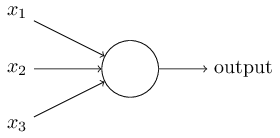


* 如果https://img-blog.csdn.net/20160720151554838 = 0 https://img-blog.csdn.net/20160720151607869 = 0，则z = -30，g(z) = 1/( 1 + e^-z )趋近于0。此外，从上图sigmoid函数的图形上也可以看出，当z=-30的时候，g(z)的值趋近于0
* 如果https://img-blog.csdn.net/20160720151554838 = 0 https://img-blog.csdn.net/20160720151607869 = 1，或https://img-blog.csdn.net/20160720151554838 =1 https://img-blog.csdn.net/20160720151607869 = 0，则z = b + https://img-blog.csdn.net/20160720151620525\*https://img-blog.csdn.net/20160720151554838 + https://img-blog.csdn.net/20160720151633098\*https://img-blog.csdn.net/20160720151607869 = -30 + 20 = -10，同样，g(z)的值趋近于0
* 如果https://img-blog.csdn.net/20160720151554838 = 1 https://img-blog.csdn.net/20160720151607869 = 1，则z = b + https://img-blog.csdn.net/20160720151620525\*https://img-blog.csdn.net/20160720151554838 + https://img-blog.csdn.net/20160720151633098\*https://img-blog.csdn.net/20160720151607869 = -30 + 20\*1 + 20\*1 = 10，此时，g(z)趋近于1。

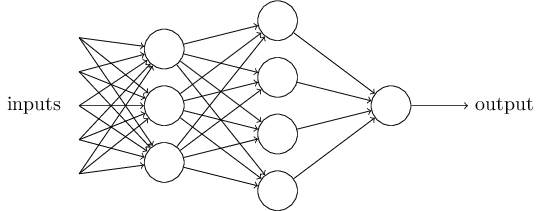
    换言之，只有https://img-blog.csdn.net/20160720151554838和https://img-blog.csdn.net/20160720151607869都取1的时候，g(z)→1，判定为正样本；https://img-blog.csdn.net/20160720151554838或https://img-blog.csdn.net/20160720151607869取0的时候，g(z)→0，判定为负样本，如此达到分类的目的。

**2.3 神经网络**

    将下图的这种单个神经元



    组织在一起，便形成了神经网络。下图便是一个三层神经网络结构

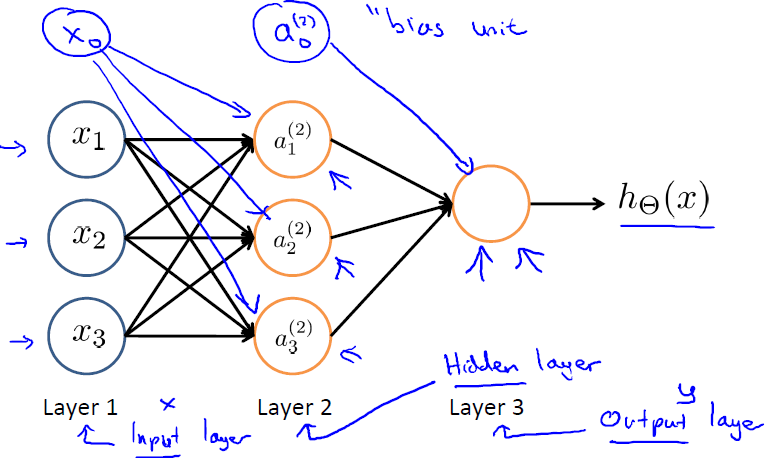


    上图中最左边的原始输入信息称之为输入层，最右边的神经元称之为输出层（上图中输出层只有一个神经元），中间的叫隐藏层。

    啥叫输入层、输出层、隐藏层呢？

* 输入层（Input layer），众多神经元（Neuron）接受大量非线形输入讯息。输入的讯息称为输入向量。
* 输出层（Output layer），讯息在神经元链接中传输、分析、权衡，形成输出结果。输出的讯息称为输出向量。
* 隐藏层（Hidden layer），简称“隐层”，是输入层和输出层之间众多神经元和链接组成的各个层面。如果有多个隐藏层，则意味着多个激活函数。

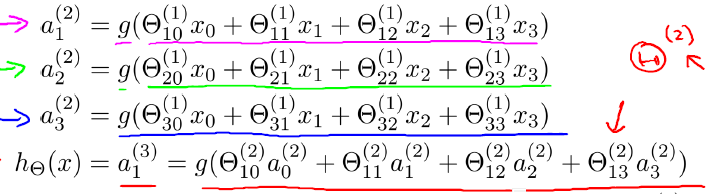
    同时，每一层都可能由单个或多个神经元组成，每一层的输出将会作为下一层的输入数据。比如下图中间隐藏层来说，隐藏层的3个神经元a1、a2、a3皆各自接受来自多个不同权重的输入（因为有x1、x2、x3这三个输入，所以a1 a2 a3都会接受x1 x2 x3各自分别赋予的权重，即几个输入则几个权重），接着，a1、a2、a3又在自身各自不同权重的影响下 成为的输出层的输入，最终由输出层输出最终结果。



    上图（图引自Stanford机器学习公开课）中

* https://img-blog.csdn.net/20160703112204318表示第j层第i个单元的激活函数/神经元
* https://img-blog.csdn.net/20160703112227254表示从第j层映射到第j+1层的控制函数的权重矩阵

    此外，输入层和隐藏层都存在一个偏置（bias unit)，所以上图中也增加了偏置项：x0、a0。针对上图，有如下公式



    此外，上文中讲的都是一层隐藏层，但实际中也有多层隐藏层的，即输入层和输出层中间夹着数层隐藏层，层和层之间是全连接的结构，同一层的神经元之间没有连接。



**3 卷积神经网络之层级结构**

[cs231n](http://cs231n.github.io/convolutional-networks/#overview)课程里给出了卷积神经网络各个层级结构，如下图



    上图中CNN要做的事情是：给定一张图片，是车还是马未知，是什么车也未知，现在需要模型判断这张图片里具体是一个什么东西，总之输出一个结果：如果是车 那是什么车

    所以

* 最左边是数据输入层，对数据做一些处理，比如去均值（把输入数据各个维度都中心化为0，避免数据过多偏差，影响训练效果）、归一化（把所有的数据都归一到同样的范围）、PCA/白化等等。CNN只对训练集做“去均值”这一步。

    中间是

* CONV：卷积计算层，线性乘积 求和。
* RELU：激励层，上文2.2节中有提到：ReLU是激活函数的一种。
* POOL：池化层，简言之，即取区域平均或最大。

    最右边是

* FC：全连接层

    这几个部分中，卷积计算层是CNN的核心，下文将重点阐述。

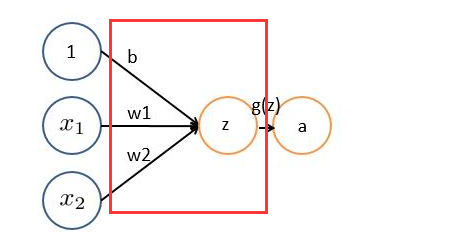
**4 CNN之卷积计算层**

**4.1 什么是卷积**

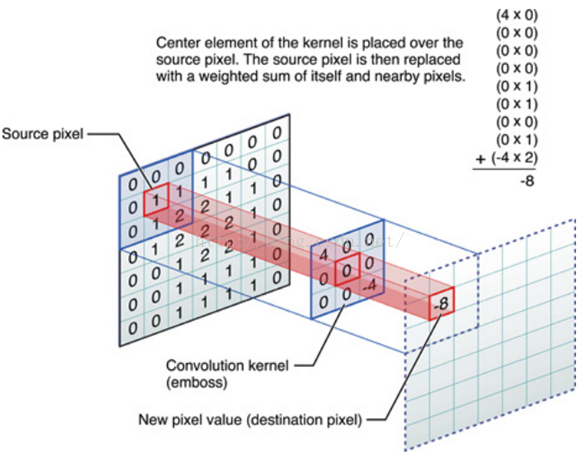
    首先，我们来了解下什么是卷积操作。

    对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器filter）做**内积**（逐个元素相乘再求和）的操作就是所谓的『卷积』操作，也是卷积神经网络的名字来源。

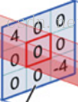
    非严格意义上来讲，下图中红框框起来的部分便可以理解为一个滤波器，即带着一组固定权重的神经元。多个滤波器叠加便成了卷积层。



    OK，举个具体的例子。比如下图中，图中左边部分是原始输入数据，图中中间部分是滤波器filter，图中右边是输出的新的二维数据。



    分解下上图

对应位置上是数字先相乘后相加=https://img-blog.csdn.net/20160702220844314

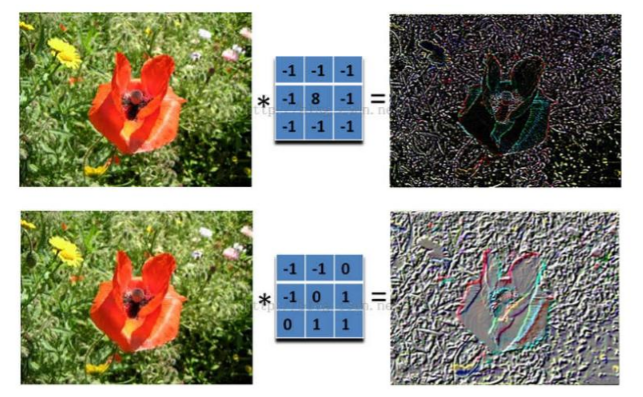
    中间滤波器filter与数据窗口做内积，其具体计算过程则是：4\*0 + 0\*0 + 0\*0 + 0\*0 + 0\*1 + 0\*1 + 0\*0 + 0\*1 + -4\*2 = -8

**4.2 图像上的卷积**

    在下图对应的计算过程中，输入是一定区域大小(width\*height)的数据，和滤波器filter（带着一组固定权重的神经元）做内积后等到新的二维数据。

    具体来说，左边是图像输入，中间部分就是滤波器filter（带着一组固定权重的神经元），不同的滤波器filter会得到不同的输出数据，比如颜色深浅、轮廓。相当于如果想提取图像的不同特征，则用不同的滤波器filter，提取想要的关于图像的特定信息：颜色深浅或轮廓。

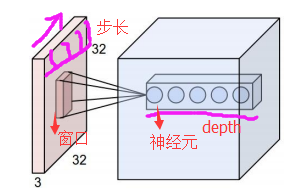
    如下图所示



**4.3 GIF动态卷积图**

    在CNN中，滤波器filter（带着一组固定权重的神经元）对局部输入数据进行卷积计算。每计算完一个数据窗口内的局部数据后，数据窗口不断平移滑动，直到计算完所有数据。这个过程中，有这么几个参数：   
　　a. 深度depth：神经元个数，决定输出的depth厚度。同时代表滤波器个数。  
　　b. 步长stride：决定滑动多少步可以到边缘。

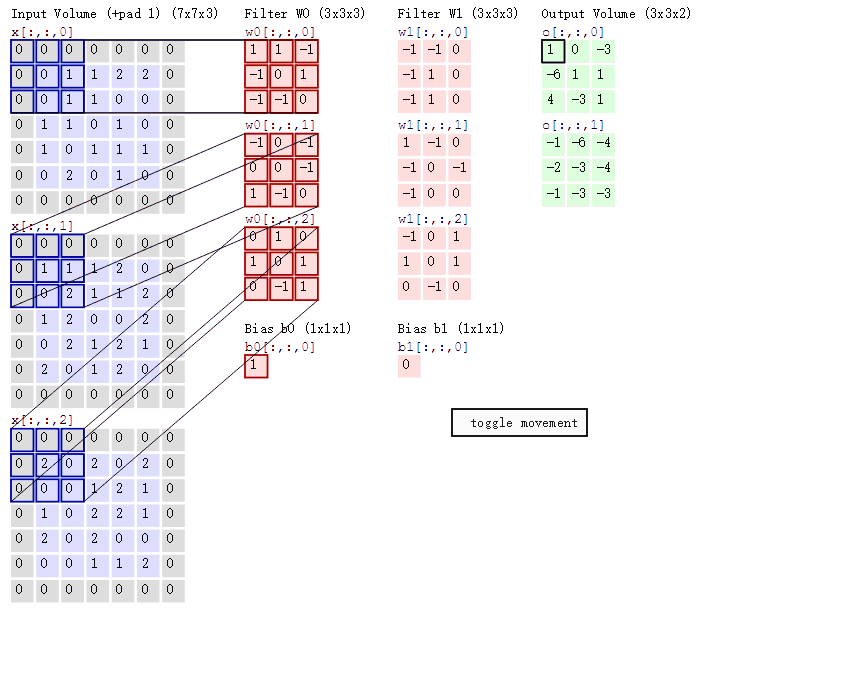
　　c. 填充值zero-padding：在外围边缘补充若干圈0，方便从初始位置以步长为单位可以刚好滑倒末尾位置，通俗地讲就是为了总长能被步长整除。



    cs231n课程中有一张卷积动图，貌似是用d3js 和一个util 画的，我**根据cs231n的卷积动图依次截取了18张图，然后用一gif 制图工具制作了一gif 动态卷积图。如下gif 图所示**，可以看到：

* 两个神经元，即depth=2，意味着有两个滤波器。
* 数据窗口每次移动两个步长取3\*3的局部数据，即stride=2。
* zero-padding=1。

    然后分别以两个滤波器filter为轴滑动数组进行卷积计算，得到两组不同的结果。



    如果初看此图，可能不一定能立马理解啥意思，但结合上文的内容后，理解这个动图已经不是很困难的事情：

* 左边是输入（**7\*7\*3**中，7\*7代表图像的像素/长宽，3代表R、G、B 三个颜色通道）
* 中间部分是两个不同的滤波器Filter w0、Filter w1
* 最右边则是两个不同的输出

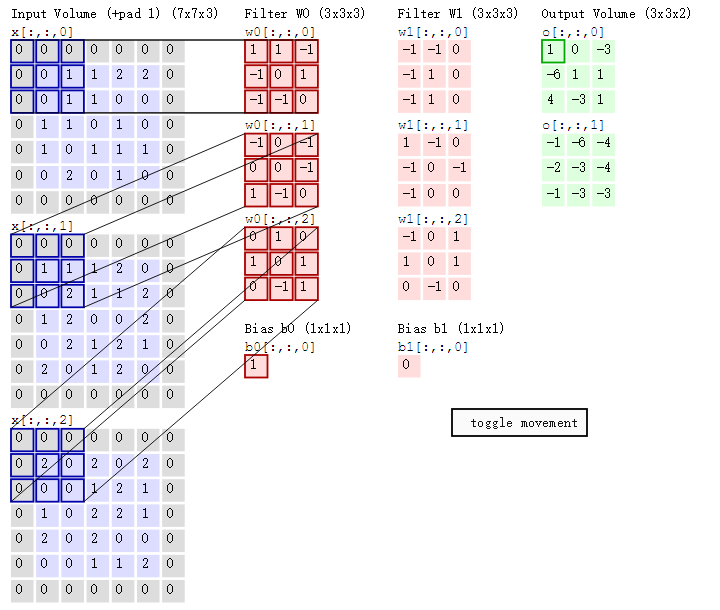
    随着左边数据窗口的平移滑动，滤波器Filter w0 / Filter w1对不同的局部数据进行卷积计算。

    值得一提的是：

1. 左边数据在变化，每次滤波器都是针对某一局部的数据窗口进行卷积，这就是所谓的CNN中的**局部感知**机制。
   * 打个比方，滤波器就像一双眼睛，人类视角有限，一眼望去，只能看到这世界的局部。如果一眼就看到全世界，你会累死，而且一下子接受全世界所有信息，你大脑接收不过来。当然，即便是看局部，针对局部里的信息人类双眼也是有偏重、偏好的。比如看美女，对脸、胸、腿是重点关注，所以这3个输入的权重相对较大。
2. 与此同时，数据窗口滑动，导致输入在变化，但中间滤波器Filter w0的权重（即每个神经元连接数据窗口的权重）是固定不变的，这个权重不变即所谓的CNN中的**参数（权重）共享**机制。
   * 再打个比方，某人环游全世界，所看到的信息在变，但采集信息的双眼不变。btw，不同人的双眼 看同一个局部信息 所感受到的不同，即一千个读者有一千个哈姆雷特，所以不同的滤波器 就像不同的双眼，不同的人有着不同的反馈结果。

    我第一次看到上面这个动态图的时候，只觉得很炫，另外就是据说计算过程是“相乘后相加”，但到底具体是个怎么相乘后相加的计算过程 则无法一眼看出，网上也没有[一目了然](https://www.baidu.com/s?wd=%E4%B8%80%E7%9B%AE%E4%BA%86%E7%84%B6&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)的计算过程。本文来细究下。

    首先，我们来分解下上述动图，如下图



    接着，我们细究下上图的具体计算过程。即上图中的输出结果1具体是怎么计算得到的呢？其实，类似wx + b，w对应滤波器Filter w0，x对应不同的数据窗口，b对应Bias b0，相当于滤波器Filter w0与一个个数据窗口相乘再求和后，最后加上Bias b0得到输出结果1，如下过程所示：



1\* 0 + 1\*0 + -1\*0

+

-1\*0 + 0\*0 + **1\*1**

+

-1\*0 + -1\*0 + 0\*1

+



-1\*0 + 0\*0 + -1\*0

+

0\*0 + 0\*1 + **-1\*1**

+

1\*0 + -1\*0 + 0\*2

+



0\*0 + 1\*0 + 0\*0

+

1\*0 + 0\*2 + 1\*0

+

0\*0 + -1\*0 + 1\*0

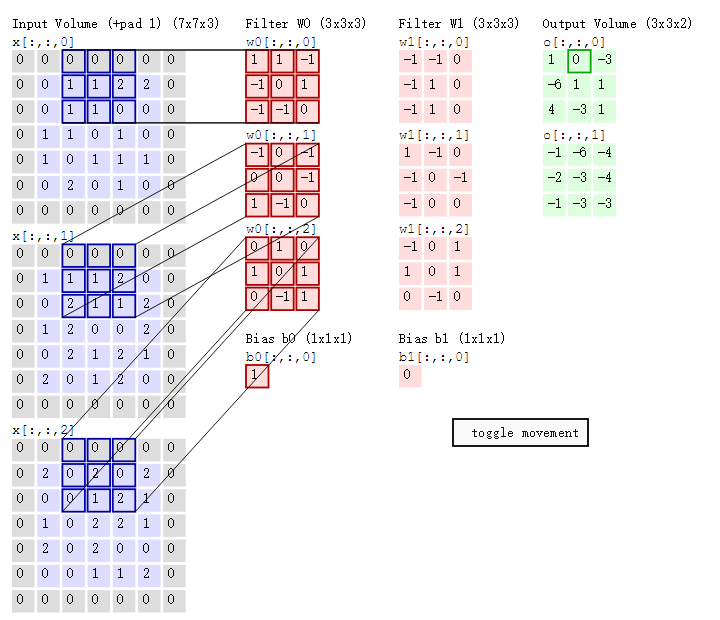
+

**1**

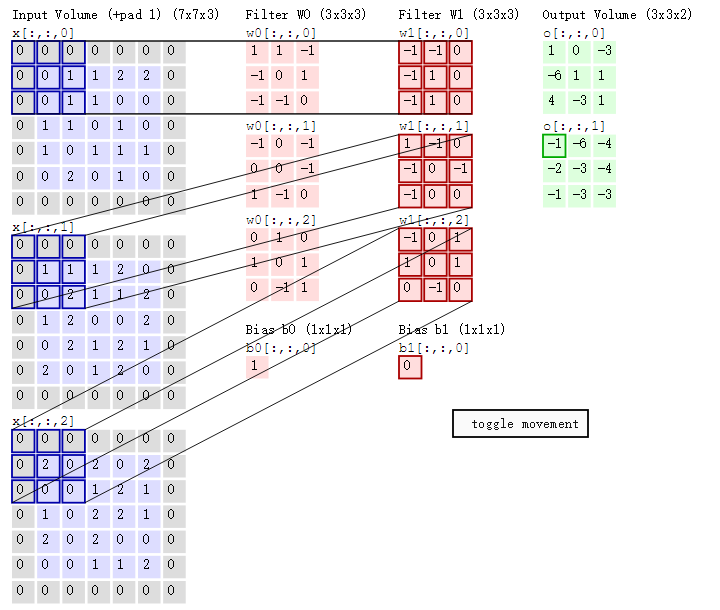
=

1

    然后滤波器Filter w0固定不变，数据窗口向右移动2步，继续做内积计算，得到0的输出结果



    最后，换做另外一个不同的滤波器Filter w1、不同的偏置Bias b1，再跟图中最左边的数据窗口做卷积，可得到另外一个不同的输出。



**5 CNN之激励层与池化层**

**5.1 ReLU激励层**

    2.2节介绍了激活函数sigmoid，但实际梯度下降中，sigmoid容易饱和、造成终止梯度传递，且没有0中心化。咋办呢，可以尝试另外一个激活函数：ReLU，其图形表示如下



    ReLU的优点是收敛快，求梯度简单。

**5.2 池化pool层**

    前头说了，池化，简言之，即取区域平均或最大，如下图所示（图引自cs231n）



    上图所展示的是取区域最大，即上图左边部分中 左上角2x2的矩阵中6最大，右上角2x2的矩阵中8最大，左下角2x2的矩阵中3最大，右下角2x2的矩阵中4最大，所以得到上图右边部分的结果：6 8 3 4。很简单不是？

**北京起宏图科技有限公司(**[**http://qhttech.com**](http://qhttech.com)**)实习招聘：**

**职位描述：从事图像识别、目标检测、物体分割等相关的CNN研发工作**

**职位要求：**

**1.计算机、电子、自动化、数学等相关专业学生**

**2.较好的编程能力**

**3.有实际使用和调试深度学习算法的经验，熟悉Caffe/Tensorflow/Torch中任一平台**

**4.熟练使用python等脚本语言，熟悉C/C++编程，熟悉linux命令**

**5.每周工作时间大于3天，连续工作6个月以上**

**6.有良好的交流沟通能力和工作态度，愿意学习和接受挑战**

**7.从事过图像识别、目标检测、物体分割等相关工作的优先考虑**

**如感兴趣请发送简历(简历命名方式:csdn+深度学习+姓名)到jingwl@bcc.ac.cn，谢谢！**

**6 参考文献及推荐阅读**

1. [人工神经网络wikipedia](http://www.wikiwand.com/zh-cn/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C)
2. [斯坦福机器学习公开课](http://52opencourse.com/139/coursera%E5%85%AC%E5%BC%80%E8%AF%BE%E7%AC%94%E8%AE%B0-%E6%96%AF%E5%9D%A6%E7%A6%8F%E5%A4%A7%E5%AD%A6%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E7%AC%AC%E5%85%AB%E8%AF%BE-%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E7%9A%84%E8%A1%A8%E7%A4%BA-neural-networks-representation)
3. <http://neuralnetworksanddeeplearning.com/>
4. 雨石 卷积神经网络：<http://blog.csdn.net/stdcoutzyx/article/details/41596663>
5. cs231n 神经网络结构与神经元激励函数：[http://cs231n.github.io/neural-networks-1/](http://cs231n.github.io/neural-networks-1/" \t "_blank)，[中译版](https://zhuanlan.zhihu.com/p/21462488" \t "_blank)
6. cs231n 卷积神经网络：[http://cs231n.github.io/convolutional-networks/](http://cs231n.github.io/convolutional-networks/" \t "_blank)
7. 七月在线寒老师讲的**5月dl班第4次课CNN与常用框架视频**，已经剪切部分放在七月在线官网：[julyedu.com](https://www.julyedu.com/video/play/42/206)
8. 七月在线5月深度学习班第5课CNN训练注意事项部分视频：[https://www.julyedu.com/video/play/42/207](https://www.julyedu.com/video/play/42/207" \t "_blank)
9. 七月在线5月深度学习班：[https://www.julyedu.com/course/getDetail/37](https://www.julyedu.com/course/getDetail/37" \t "_blank)
10. 七月在线5月深度学习班课程笔记——No.4《CNN与常用框架》：[http://blog.csdn.net/joycewyj/article/details/51792477](http://blog.csdn.net/joycewyj/article/details/51792477" \t "_blank)
11. 七月在线6月数据数据挖掘班第7课视频：数据分类与排序
12. 手把手入门神经网络系列(1)\_从初等数学的角度初探神经网络：[http://blog.csdn.net/han\_xiaoyang/article/details/50100367](http://blog.csdn.net/han_xiaoyang/article/details/50100367" \t "_blank)
13. 深度学习与计算机视觉系列(6)\_神经网络结构与神经元激励函数：[http://blog.csdn.net/han\_xiaoyang/article/details/50447834](http://blog.csdn.net/han_xiaoyang/article/details/50447834" \t "_blank)
14. 深度学习与计算机视觉系列(10)\_细说卷积神经网络：[http://blog.csdn.net/han\_xiaoyang/article/details/50542880](http://blog.csdn.net/han_xiaoyang/article/details/50542880" \t "_blank)
15. zxy 图像卷积与滤波的一些知识点：[http://blog.csdn.net/zouxy09/article/details/49080029](http://blog.csdn.net/zouxy09/article/details/49080029" \t "_blank)
16. zxy 深度学习CNN笔记：[http://blog.csdn.net/zouxy09/article/details/8781543/](http://blog.csdn.net/zouxy09/article/details/8781543/" \t "_blank)
17. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>，[中译版](http://www.csdn.net/article/2015-11-11/2826192" \t "_blank)
18. 《神经网络与深度学习》中文讲义：[http://vdisk.weibo.com/s/A\_pmE4iIPs9D](http://vdisk.weibo.com/s/A_pmE4iIPs9D" \t "_blank)
19. ReLU与sigmoid/tanh的区别：<https://www.zhihu.com/question/29021768>
20. CNN、RNN、DNN内部网络结构区别：<https://www.zhihu.com/question/34681168>
21. 理解卷积：<https://www.zhihu.com/question/22298352>
22. 神经网络与深度学习简史：[1 感知机和BP算法](http://mp.weixin.qq.com/s?__biz=MzA3MzI4MjgzMw==&mid=402032673&idx=1&sn=d7e636b6d033cbcf8a74dfaf710e9ccf&scene=21#wechat_redirect)、[4 深度学习的伟大复兴](http://chuansong.me/n/2523251)
23. 在线制作gif 动图：<http://www.tuyitu.com/photoshop/gif.htm>
24. [支持向量机通俗导论（理解SVM的三层境界）](http://blog.csdn.net/v_july_v/article/details/7624837)