卷积神经网络讲义

左谭励 2017.12.3

主要内容

- 卷积（由来，内涵，外延）
- 图像处理的滤波（仅介绍空间滤波）
- 神经网络的定义（以及优化方法）
- 现代CNN的例子（LeNet，一些前沿的结构）

卷积与信号处理的关系

对于一个线性时不变系统 $y(t) = H[x(t)]$，输出和输入是有着某种关系的。尝试找到其数学表达式，就可以得到卷积的定义。

> 我们称一个系统是线性的，当且仅当 $H[ax + by] = aH[x] + bH[y]$。
>
> 一个时不变系统满足 $H[x(t)] = y(t) \Rightarrow H[x(t-a)] = y(t-a)$。
>
> 满足这两个性质的系统有放大器：$y(t) = kx(t)$，差分机 $y(t) = x'(t)$ 。

定义冲激函数（又称狄拉克δ函数）为：

$$f(x) = \begin{cases} +\infty & t = 0 \\ 0 & t \neq 0 \end{cases}$$

且满足$\int_{-\infty}^{+\infty} \delta(t)dt = 1$此时，$x(t) = \int x(\tau)\delta(t-\tau)d\tau$

定义 $h(t) = H[\delta(t)]$ ，也就是系统对冲激函数的响应。

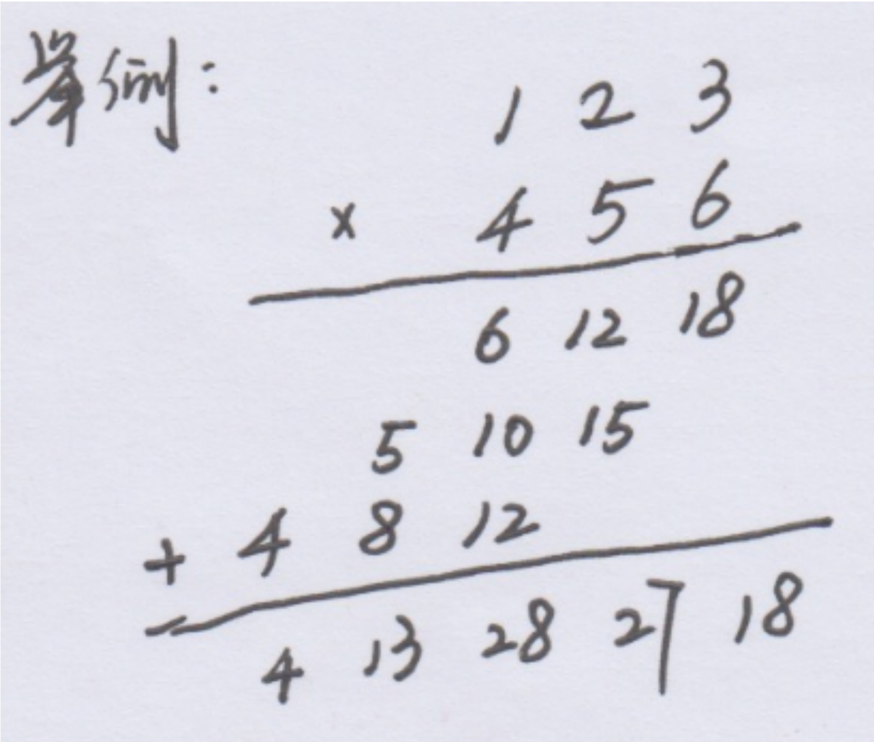
于是就得到了卷积的数学定义：$x(t) * h(t) = \int x(\tau)\delta(t-\tau)dt$，它的一个含义就是线性时不变系统的数学解析。

卷积举例：整除乘法与离散卷积

离散情况下卷积的定义为 $y[n] = \sum x[k]h[n-k]$，离散情况下 $\delta[0] = 1, \delta[n] = 0 (n \neq 0)$。

整数A, B在十进制下可表示为 $A = \sum 0^{\infty} a_k \cdot 10^k, B = \sum 0^{\infty} b_k \cdot 10^k$，那么 $A \cdot B = \sum 0^{\infty} a_k \cdot bn - k \cdot 10^k$

如果列出竖式，不考虑进位（如下图），那么每一列就是卷积的某一项。



另一种计算方法

离散卷积也可以这样来计算：

1. 将其中一个序列转置 ($x[n] = x[-n]$)

2. 将其中一个序列固定，平移另一个序列，每次将对应位置的数字相乘然后求和。

卷积举例：差分

离散的差分定义为 $y[n] = x[n] - x[n-1]$. 很明显，我们可以用 h[n] = {1, -1, 0, 0....} 作为卷积核就可以了。实际上，h[n] 就是 $\delta[n]$ 的一阶差分。更一般的，对 $\delta[n]$ 的 k 阶差分和 x 做卷积，就可以得到 x 的 k 阶段差分。

二维情况下的差分

定义：$y[n, m] = \sum_x \sum_y X[x, y]h[n-x, n-y]$. 二维情况下也可以看作将卷积核心转置之后，一步一步平移来得到计算结果。对一个矩阵的二维差分也可以先对 δ 函数差分再卷积。

图像处理中常用的卷积操作

在图像处理中，图像被看作是一个矩阵。对于灰度图像，每个点的值表示一个像素的强度，彩色图像会将每个像素分解成RGB三个通道。

数字图像处理领域经常食用一些滤波的操作，而大部分的滤波都可以表示成一个卷积。这其中包括对图像进

行平滑（模糊）、锐化（增强）、边缘提取等。用于这些卷积操作的矩阵称作卷积核。

平滑滤波

在一维情况下，我们要平滑一个函数常见的方法是求平均值。下面的几个卷积本质上也是在领域内计算某种加权平均。

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

如果要达到更强的模糊效果，可以考虑更大的领域（用更大的卷积核，例如7x7, 11x11）。

锐化滤波

对图像锐化实际上是增强细节。一般这么构造：某种模糊滤波 G[A]，A - G[A] 得到细节，然后加到原图上去，表达式为：$H[A] = A + \alpha(A - G[A])$。

下面是几个例子：

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

边缘检测

图像中的边缘在矩阵上的表示就是某个方向灰度的突变，这种突变可以用差分来求解。下面是几个例子：

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

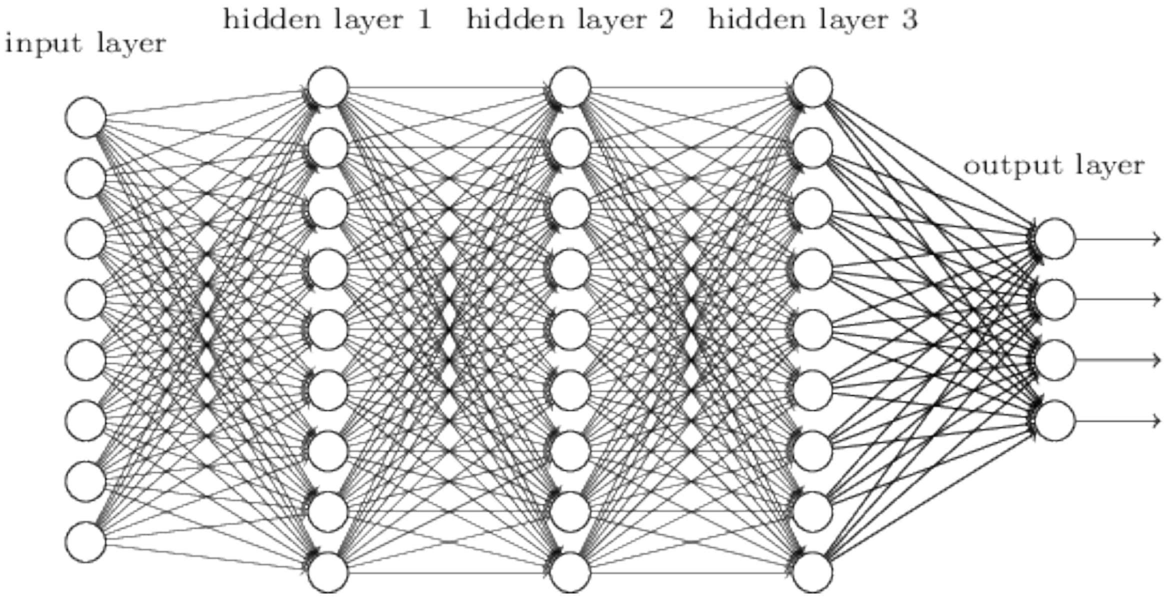
更多的滤波的例子可以参照：http://www.jianshu.com/p/cbd1a1f86d1b

传统神经网络

伸进网络本质上是一些数学模型，定了对输入 x 和参数 θ 做了怎样的运算得到输出值，也就是一个函数 $f(x;\theta)$ 。标记：$x^{(i)}$ 是第 i 个数据的输入，$y^{(i)}$ 是第 i 个数据的输出。通过选取合适的 θ 使得对于所有的数据，$f(x^{x(i)};\theta)$ 尽量接近 $y^{(i)}$. 为了度量这个“接近”的含义，我们可以选取一些函数，例如均方差 $\frac{1}{m}\sum_i |f(x^{x(i)};\theta) - y^{(i)}|^2$，这个值越小说明接近程度越高。我们称这样的函数为 loss 函数。

因为而 x 和 y 是固定的，我们只需要优化 θ 使得 loss 尽量小，这样 loss 函数就可以看作是一个关于 θ 的函数，记作 $J(\theta) = loss(f(X;\theta), Y)$。

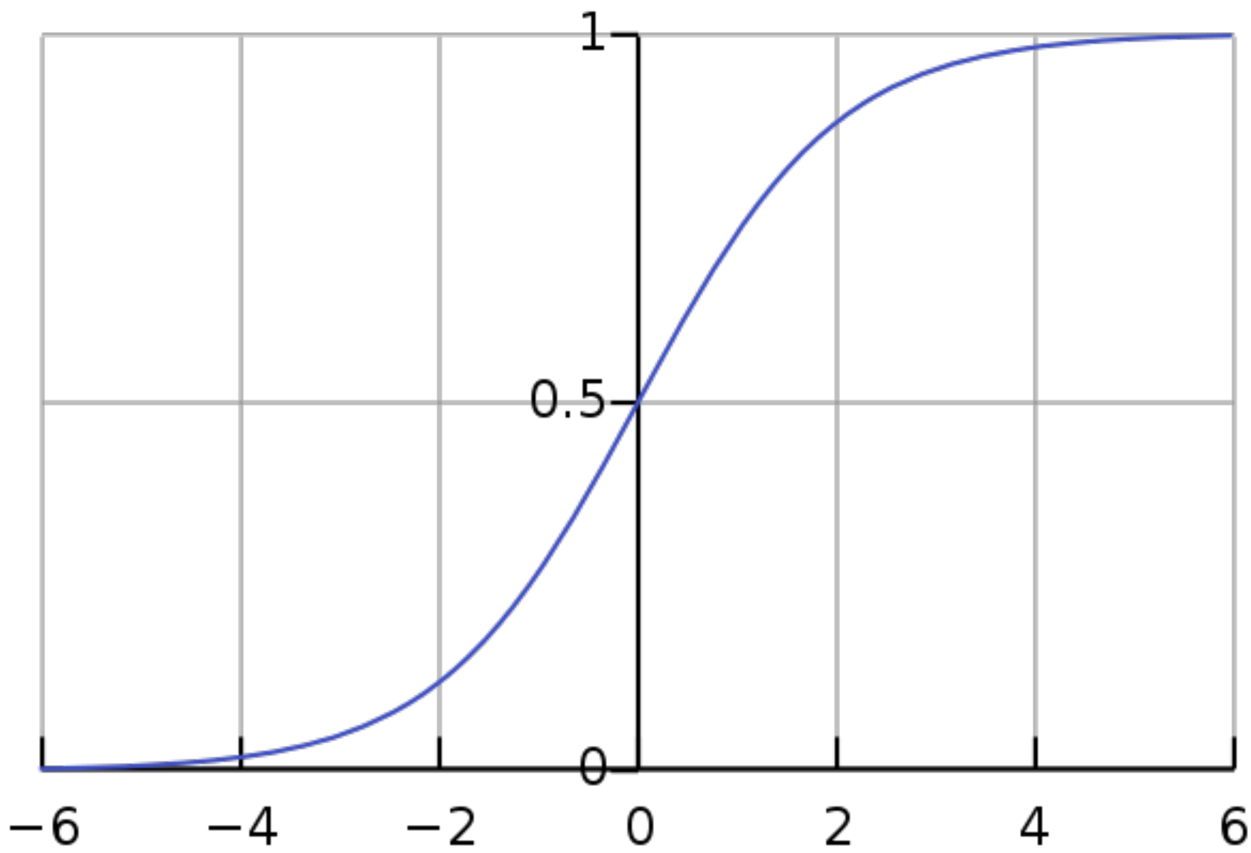
神经网络的定义



上图是一个卷积神经网络的例子。第一列是输入层，之后有三个隐藏层，最后计算的结果输出到输出层。对于单个神经元（节点）来说，其计算规则是对之前和它相连的所有神经元的输出做一个加权求和，然后通过某个激活函数确定输出的值。$y = \sigma(\sum_i w_i x_i + b)$ 其中 w_i 为第 i 个连接的权值，x_i 是这个神经元第 i 个输入神经元，b 是 bias，σ是某个非线性函数。

讨论：如果$sigma$是线形函数，会导致什么结果？这样的话不管网络有多深，这个网络锁代表的一个函数一定是关于 x 的一个线性变换。

通常会选取的函数是逻辑函数，$\sigma(x) = \frac{1}{1+e^{-x}}$ 。



另一个经常使用的激活函数是ReLU $f(x) = x^+ = \max(0, x)$, 它在被 Alex 首次提出之后，成为了现代神经网络的标配。至于它为何会带来改进，后面再展开。

用 $\overrightarrow{h^{(1)}}, \overrightarrow{h^{(2)}}, \overrightarrow{h^{(3)}}$ 分别表示三个隐藏层，x, y表示输入和输出层，
$\boldsymbol{W^{(1)}}, \boldsymbol{W^{(2)}}, \boldsymbol{W^{(3)}}, \boldsymbol{W^{(4)}}, \overrightarrow{b^{(1)}}, \overrightarrow{b^{(2)}}, \overrightarrow{b^{(3)}}, \overrightarrow{b^{(4)}}$ 为层之间电系数，那么上面的神经网络定义了这样的一种运算：

$$\overrightarrow{h^{(1)}} = \sigma(\boldsymbol{W^{(1)}}\vec{x} + \overrightarrow{b^{(1)}})$$
$$\overrightarrow{h^{(2)}} = \sigma(\boldsymbol{W^{(2)}}\overrightarrow{h^{(1)}} + \overrightarrow{b^{(2)}})$$
$$\overrightarrow{h^{(3)}} = \sigma(\boldsymbol{W^{(3)}}\overrightarrow{h^{(2)}} + \overrightarrow{b^{(3)}})$$
$$\vec{y} = g(\boldsymbol{W^{(4)}}\overrightarrow{h^{(3)}} + \overrightarrow{b^{(4)}})$$

神经网络的优化

之前说过，将神经网络当作一个函数，那么我们的目标就是优化参数θ使得loss函数最小。假定我们选择 $L(\cdot)$ 作为 loss 函数，那么 $J(\theta) = \sum_i L(f(x^{(i)}; \theta), y^{(i)})$。

通常我们使用随机梯度下降的方法来优化这个 θ。不断迭代这样的过程：

1. 选择一批数据X，求出梯度 $\partial J(\theta)/\partial\theta$。（因为迭代次数可能很多，所以不可能每次都计算完所有数据）
2. 用 $\theta - \eta\frac{\partial J(\theta)}{\partial\theta}$ 来更新 θ。

我们尝试推导一下W的梯度

$$\frac{\partial J}{\partial W^{(4)}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial W^{(4)}}$$

$$\frac{\partial J}{\partial W^{(3)}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial h^{(4)}}\frac{\partial h^{(4)}}{\partial W^{(3)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial h^{(4)}}\frac{\partial h^{(4)}}{\partial h^{(3)}}\frac{\partial h^{(3)}}{\partial W^{(2)}}$$

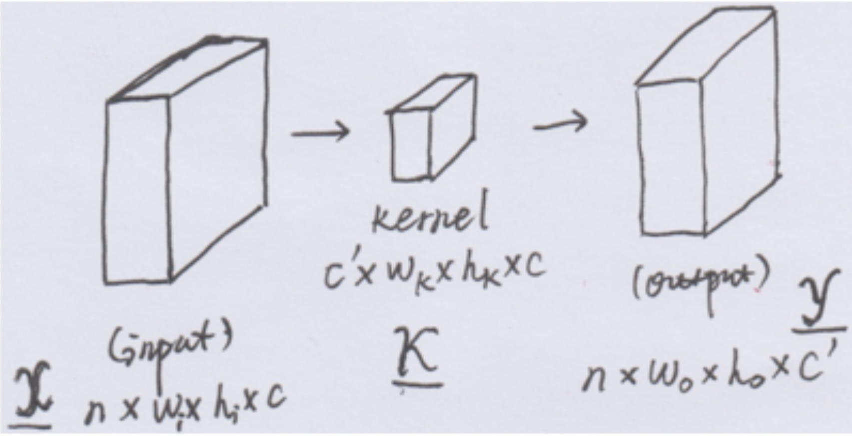
$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial h^{(4)}}\frac{\partial h^{(4)}}{\partial h^{(3)}}\frac{\partial h^{(3)}}{\partial h^{(2)}}\frac{\partial h^{(2)}}{\partial W^{(1)}}$$

上面的式子中用到了链式法则，而每一个 h 对 h 或者 h 对 W 的偏导都要乘上 σ' 。当我们使用逻辑函数作为激活函数的时候，很容易因为某一项接近于0导致全部为0，而用 ReLU 函数就不会有这个问题。

现代 CNN 的构成

卷积层

之前我们提到了，卷积神经网络就是一个运算。那么我们可以用卷积进行计算。



输入是一个 $N \times w_i \times h_i \times c$ 的矩阵，卷积核大小是 $c' \times w_k \times h_k \times c$，输出大小是$N \times w_o \times h_o \times c'$。

其中N是 batch 大小，$w_i \times h_i$ 是输入的每一个图像的长宽，$w_o \times h_o$ 是输出的图像的长宽。c 是通道大小。对于第一个输入层来说，c 通常是1（黑白）或者3（彩色），而后面的层的通道数是人为定的。

这个卷积层的意义是将每个 feature map 做一次扩展的卷积，用 c' 个不同的卷积核得到输出图像不同通道的数据。对于每个卷积核，如果不考虑 batch 维，输入和输出的关系如下：

$$\mathcal{Y}[x, y, z] = \sum_i \sum_j \sum_k \mathcal{X}[i, j, k] \cdot \mathcal{K}[z, x-i, y-j, k]$$

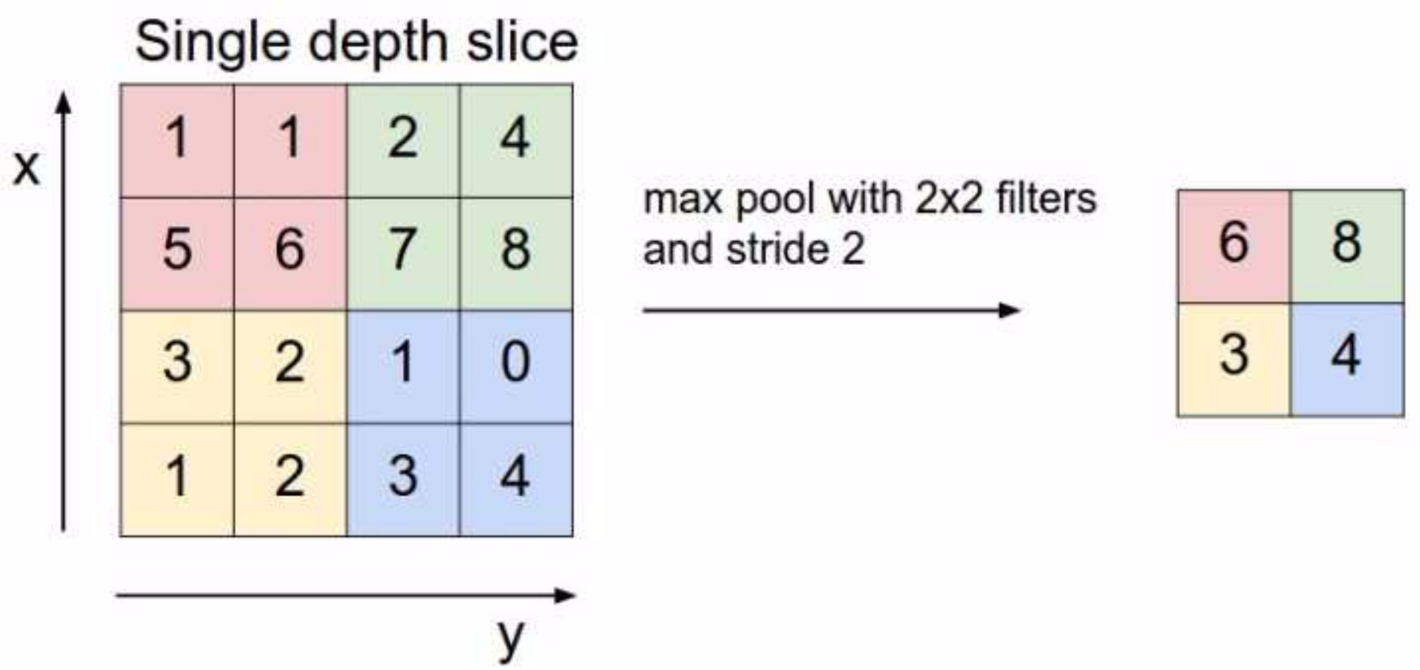
大致上和二维卷积差不多，就是在通道（深度）维上用向量的点积代替了数字乘。

输入和输出具体的参数关系如下图（参见 http://cs231n.github.io/convolutional-networks/）

- 输入尺寸：W_1 x H_1 x D_1
- 卷积层参数：
  - filters个数：K
  - filters尺寸：F
  - stride：S
  - zero padding： P
- 输出尺寸：W_2 x H_2 x D_2，其中
  - $W_2 = (W_1 + 2P - F)/S + 1$
  - $H_2 = (H_1 + 2P - F)/S + 1$
  - $D_2 = K$
- 权重个数：权重共享后
  - 每个filter有$F \cdot F \cdot D_1$
  - 一共有$F \cdot F \cdot D_1 \cdot K$个权重及$K$个偏移。
- 常规设置：$F = 3, S = 1, P = 1$

池化层

将整个图片被重叠（或者不重叠）的分割成若干个同样大小的小块（pooling size），每个小块内内进行下采样，就得到了输出。



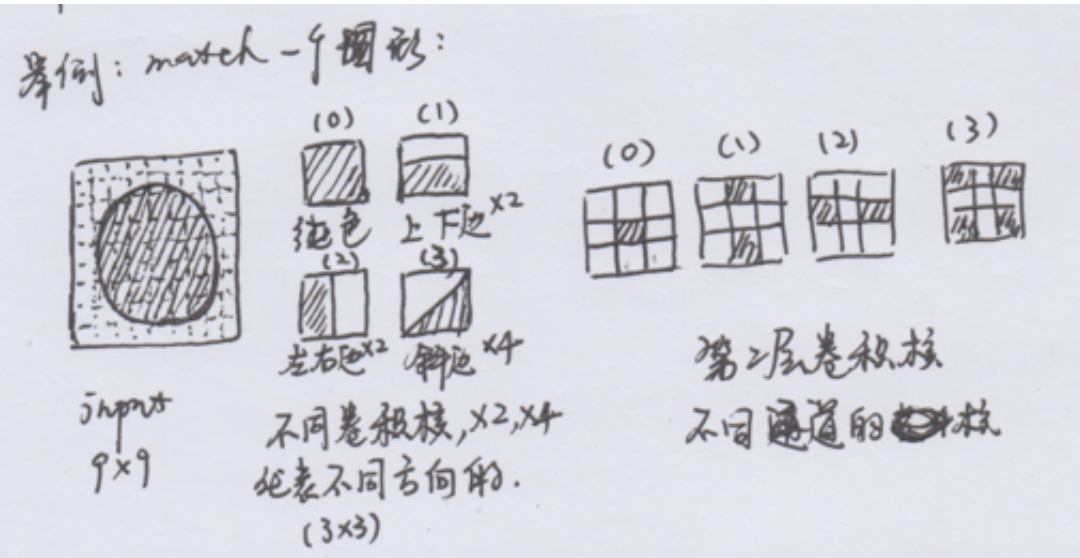
通常下采样的方法是取最大值(max-pooling)，如上图。这个层的主要功能是下采样，减少变量个数。

对卷积网络结构的一些思考

主流观点认为卷积+池化的主要目的是减少参数。卷积层用了共享的参数，相对于全连接（也就是之前介绍的传统神经网络）大大降低了参数使用量。从另一个角度来说，卷积也是经典算法中常用的操作。那么神经网络训练的过程可以看作是在学一种卷积核。

另一种解释是认为这是模式匹配的一种方法。每一次卷积可以拆分为多次点积，而点积又可以理解为一个区域和 kernel 的相似度（$u \cdot v = |u| \cdot |v| sin\theta$，$\theta$是两个向量的夹角。夹角越小，$sin\theta$越大，说明相似度越高）。而 max-pooling 的目的就是在一个范围内选取一个最大相似度来将维。

举个例子：



浅层的卷积学到了不同方向的边缘、纯色等特征，第二层的一个核通过第一层的特征聚合成一种圆的判别器。