卷积与互相关

给定一个图像$X \in \mathbb{R}^{M\times N}$，和滤波器 $W \in \mathbb{R}^{m\times n}$，一般$m \ll M$, $n \ll N$，其卷积为

$$y_{ij} = \sum_{u=1}^{m}\sum_{v=1}^{n} w_{uv} \cdot x_{i-u+1,j-v+1}. \quad \begin{matrix} m\leqslant i\leqslant M \\ n\leqslant j\leqslant N \end{matrix}$$

它们的互相关为

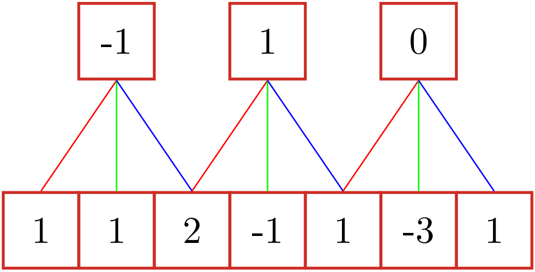
$$y_{ij} = \sum_{u=1}^{m}\sum_{v=1}^{n} w_{uv} \cdot x_{i+u-1,j+v-1}. \quad \begin{matrix} 1\leqslant i\leqslant M\text{-}m+1 \\ 1\leqslant j\leqslant N\text{-}n+1 \end{matrix}$$

互相关和卷积的区别在于卷积核仅仅是否进行翻转（卷积窗口在数据上移动时，在一个窗口内，权参数的尾部与数据头部相乘，权参数的头部与数据尾部相乘，等价于先将窗口翻转，然后头与头相乘，尾与尾相乘）。因此互相关也可以称为不翻转卷积。

翻转就是从两个维度（从上到下、从左到右）颠倒次序，即旋转 180 度。也可以理解为图像是否进行翻转。

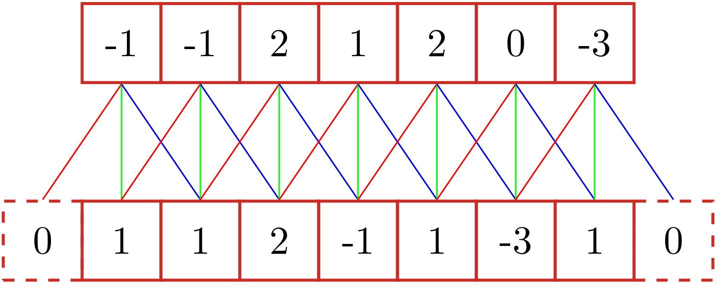
卷积的变种

滤波器的步长是指滤波器在滑动时的时间间隔。



步长$s = 2$

零填充（zero padding）是在输入向量两端进行补零。



零填充$p = 1$

假设卷积层的输入神经元个数为n，卷积大小为m，步长（stride）为s，输入神经元两端各填补p个零（zero padding），那么该卷积层的神经元数量为$(n-m+2p)/s+1$。

一般常用的卷积有以下三类：

- 窄卷积（narrow convolution）：步长$s=1$，两端不补零$p=0$，卷积后输出长度为$n-m+1$。默认
- 宽卷积（wide convolution）：步长$s=1$，两端补零$p=m-1$，卷积后输出长度$n+m-1$。
- 等长卷积（equal-width convolution）：步长$s=1$，两端补零$p=(m-1)/2$，卷积后输出长度n。

卷积的数学性质

交换性

不限制两个卷积信号的长度，具有交换性，即 $\mathbf{x} \otimes \mathbf{y} = \mathbf{y} \otimes \mathbf{x}$。
当输入信息和卷积核有固定长度时，宽卷积依然具有交换性。

图像 $X \in \mathbb{R}^{M\times N}$ 和卷积核 $W \in \mathbb{R}^{m\times n}$，对 X 两个维度进行零填充，两端各补 $m-1$ 和 $n-1$ 个零，得到全填充（full padding）的图像 $\tilde{X} \in \mathbb{R}^{(M+2m-2)\times(N+2n-2)}$。图像 X 和卷积核 W 的宽卷积（wide convolution）定义为

$$W \tilde{\otimes} X \triangleq W \otimes \tilde{X},$$

宽卷积具有交换性，即 $\mathbf{rot180}(W) \tilde{\otimes} X = \mathbf{rot180}(X) \tilde{\otimes} W.$

假设 $Y = W \otimes X$, 其中 $X \in \mathbb{R}^{M\times N}$, $W \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{(M-m+1)\times(N-n+1)}$, 函数 $f(Y) \in \mathbb{R}$ 为一个标量函数，则

$$y_{ij} = \sum_{k,q} w_{kq}\, x_{i+k-1,j+q-1}$$

$$\begin{aligned}
\frac{\partial f(Y)}{\partial w_{uv}} &= \sum_{i=1}^{M-m+1}\sum_{j=1}^{N-n+1} \frac{\partial y_{ij}}{\partial w_{uv}} \frac{\partial f(Y)}{\partial y_{ij}} \\
&= \sum_{i=1}^{M-m+1}\sum_{j=1}^{N-n+1} x_{i+u-1,j+v-1} \frac{\partial f(Y)}{\partial y_{ij}} \\
&= \sum_{i=1}^{M-m+1}\sum_{j=1}^{N-n+1} \frac{\partial f(Y)}{\partial y_{ij}} x_{u+i-1,v+j-1}
\end{aligned}$$

可以看出，$f(Y)$ 关于 W 的偏导数为 X 和 $\frac{\partial f(Y)}{\partial Y}$ 的卷积

$$\frac{\partial f(Y)}{\partial W} = \frac{\partial f(Y)}{\partial Y} \otimes X. \qquad \boxed{\text{公式1}}$$

同理得到，

$y_{ij} = \sum_{k,q} w_{kq}\, x_{i+k-1,j+q-1}$

令 $s=i+k-1, t=j+q-1$

则 $k=s-i+1, q=t-j+1$

$$\frac{\partial f(Y)}{\partial x_{st}} = \sum_{i=1}^{M-m+1}\sum_{j=1}^{N-n+1} \frac{\partial y_{ij}}{\partial x_{st}} \frac{\partial f(Y)}{\partial y_{ij}}$$
$$= \sum_{i=1}^{M-m+1}\sum_{j=1}^{N-n+1} w_{s-i+1,t-j+1} \frac{\partial f(Y)}{\partial y_{ij}},$$

其中当 $(s-i+1)<1$，或 $(s-i+1)>m$，或 $(t-j+1)<1$，或 $(t-j+1)>n$ 时，$w_{s-i+1,t-j+1}=0$。即相当于对 W 进行了 $p=(M-m, N-n)$ 的零填充。

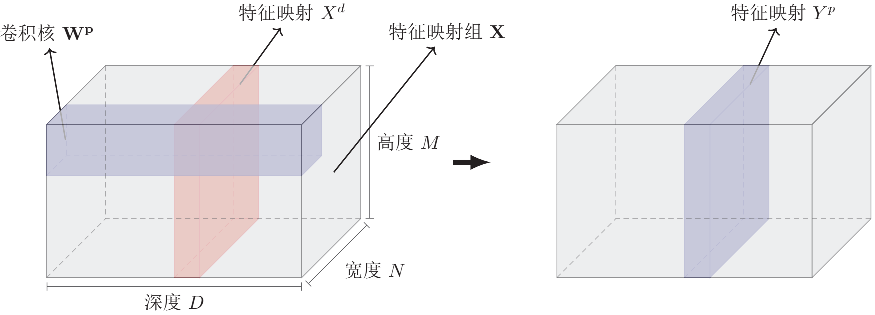
可以看出，$f(Y)$ 关于 X 的偏导数为 W 和 $\frac{\partial f(Y)}{\partial Y}$ 的宽卷积。公式中的卷积是真正的卷积而不是互相关，为了一致性，用互相关的 “卷积”，即

$$\frac{\partial f(Y)}{\partial X} = \mathbf{rot180}\left(\frac{\partial f(Y)}{\partial Y}\right)\tilde{\otimes} W$$
$$= \mathbf{rot180}(W)\tilde{\otimes}\frac{\partial f(Y)}{\partial Y},$$

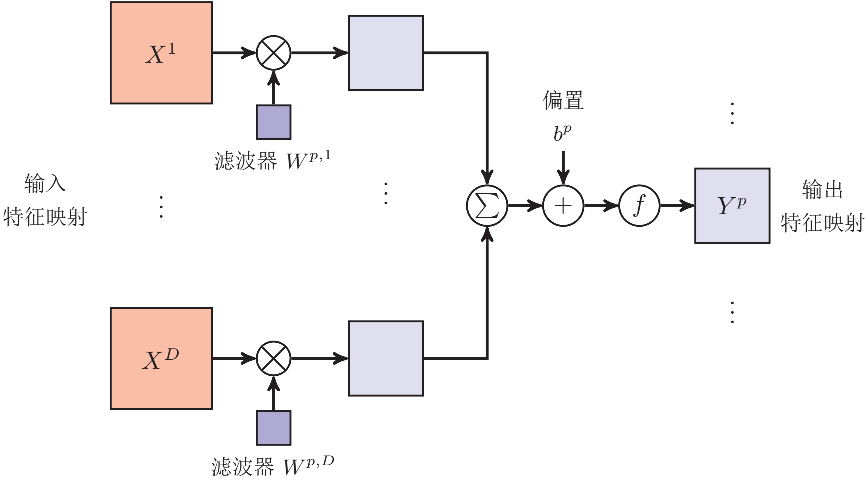
公式2

其中 $\mathbf{rot180}(\cdot)$ 表示旋转 180 度。

卷积层的一般结构：



- 输入特征映射组：$\mathbf{X} \in \mathbb{R}^{M\times N\times D}$ 为三维张量（tensor），其中每个切片 (slice) 矩阵 $X^d \in \mathbb{R}^{M\times N}$ 为一个输入特征映射，$1 \le d \le D$；
- 输出特征映射组：$\mathbf{Y} \in \mathbb{R}^{M'\times N'\times P}$ 为三维张量，其中每个切片矩阵 $Y^p \in \mathbb{R}^{M'\times N'}$ 为一个输出特征映射，$1 \le p \le P$；
- 卷积核：$\mathbf{W} \in \mathbb{R}^{m\times n\times D\times P}$ 为四维张量，其中每个切片矩阵 $W^{p,d} \in \mathbb{R}^{m\times n}$ 为一个两维卷积核，$1 \le d \le D, 1 \le p \le P$。



卷积层中从输入特征映射组 $\mathbf{X}$ 到输出特征映射 Y^p 的计算

$$
\begin{aligned}
Z^p &= \mathbf{W}^p \otimes \mathbf{X} + b^p = \sum_{d=1}^{D} W^{p,d} \otimes X^d + b^p, \\
Y^p &= f(Z^p).
\end{aligned}
$$

$\mathbf{W}^p \in \mathbb{R}^{m\times n\times D}$：三维卷积核

$f(\cdot)$：非线性激活函数

在输入为$\mathbf{X} \in \mathbb{R}^{M\times N\times D}$，输出为$\mathbf{Y} \in \mathbb{R}^{M'\times N'\times P}$的卷积层中，每一个输出特征映射都需要$D$个滤波器以及一个偏置。假设每个滤波器的大小为$m \times n$，那么共需要$P \times D \times (m \times n) + P$个参数。

汇聚层

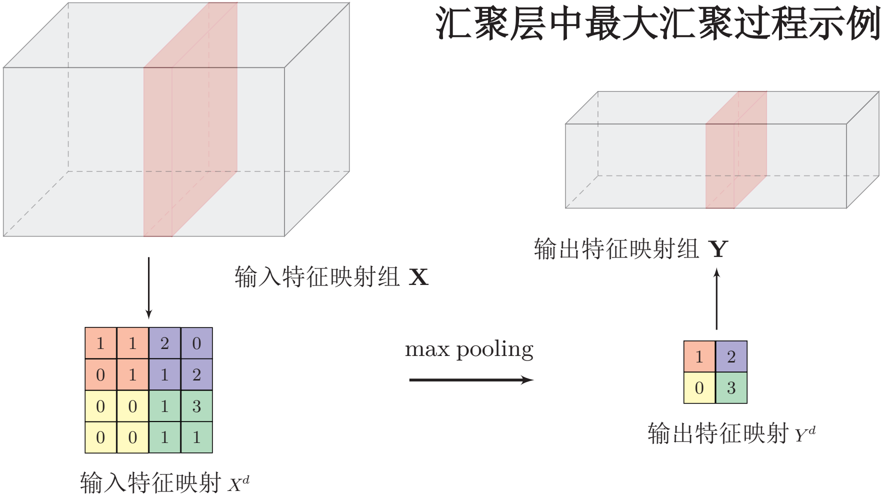
汇聚层（pooling layer）也叫子采样层，其作用是进行特征选择，降低特征数量，并从而减少参数数量。减少特征维数也可以通过增加卷积步长来实现。

假设汇聚层的输入特征映射组为$\mathbf{X} \in \mathbb{R}^{M\times N\times D}$，对每个特征映射$X^d$，将其划分为很多区域$R^d_{m,n}, 1 \le m \le M', 1 \le n \le N'$，$m$、$n$是区域标号。这些区域可以重叠，也可以不重叠。汇聚是指对每个区域进行下采样得到一个值，作为这个区域的概括。主要包括最大汇聚和平均汇聚。

对每一个输入特征映射X^d的$M' \times N'$个区域进行子采样，得到汇聚层输出特征映射$Y^d = \{Y^d_{m,n}\}, 1 \le m \le M', 1 \le n \le N'$。

汇聚层不但可以有效地减少神经元的数量，还可以使得网络对一些小的局部形态改变保持不变性，并拥有更大的感受野。

汇聚层中最大汇聚过程示例

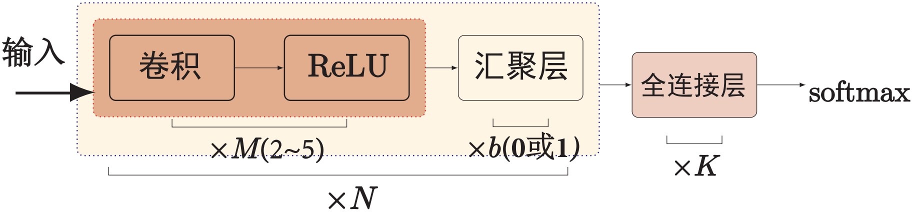


目前主流的卷积网络中，汇聚层仅包含下采样操作。但在早期的一些卷积网络（比如LeNet-5）中，有时也会在汇聚层使用非线性激活函数，比如

$$Y'^d = f\left(w^d \cdot Y^d + b^d\right),$$

其中 Y'^d 为汇聚层的输出，$f(\cdot)$ 为非线性激活函数，w^d 和 b^d 为可学习的标量权重和偏置。

典型的卷积网络结构



目前，整个网络结构趋向于使用更小的卷积核（比如1 × 1和3 × 3）以及更深的结构（比如层数大于50）。

参数学习

第 l 层为卷积层，作为其输入的特征映射为 $\mathbf{X}^{(l-1)} \in \mathbb{R}^{M\times N\times D}$

第 l 层特征映射净输入 $\mathbf{Z}^{(l)} \in \mathbb{R}^{M'\times N'\times P}$，第 l层的第 $p(1 \leq p \leq P)$ 个特征映射净输入

$$Z^{(l,p)} = \sum_{d=1}^{D} W^{(l,p,d)} \otimes X^{(l-1,d)} + b^{(l,p)} \mathbf{1}_{M'\times N'}$$

$W^{(l,p,d)}$ 和 $b^{(l,p)}$ 为卷积核以及偏置。第 l 层中共有 $P \times D$ 个卷积核和 P 个偏置。

$$\frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial W^{(l,p,d)}} = \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial Z^{(l,p)}} \otimes X^{(l-1,d)} \quad \boxed{\text{利用公式1}}$$

$$= \delta^{(l,p)} \otimes X^{(l-1,d)},$$

其中 $\delta^{(l,p)}$ 为损失函数关于第 l 层的第 p 个特征映射净输入 $Z^{(l,p)}$ 的偏导数。

损失函数关于第 l 层的第 p 个偏置 $b^{(l,p)}$ 的偏导数为

$$\frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial b^{(l,p)}} = \sum_{i,j} [\delta^{(l,p)}]_{i,j}$$

卷积网络中，每层参数的梯度依赖其所在层的误差项 $\delta^{(l,p)}$。

误差项的计算

卷积层和汇聚层的误差项计算有所不同

汇聚层 第 $l+1$ 层为汇聚层，第 l 层的第 p 个特征映射的误差项：

$$\delta^{(l,p)} \triangleq \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial Z^{(l,p)}} = \frac{\partial X^{(l,p)}}{\partial Z^{(l,p)}} \cdot \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial X^{(l,p)}} \quad \boxed{\text{公式3}}$$

$$= f'(Z^{(l,p)}) \odot \mathbf{up}(\delta^{(l+1,p)})$$

卷积层 $l+1$ 为卷积层，特征映射净输入 $\mathbf{Z}^{(l+1)} \in \mathbb{R}^{M'\times N'\times K}$，

第 $k(1 \leq k \leq K)$ 个特征映射净输入

$$Z^{(l+1,k)} = \sum_{p=1}^{P} W^{(l)/,i^{*}p)} \otimes X^{(l,n)} + b^{(l)/,i)}\, 1_{M'\times N'}$$

第 $l+1$ 层中共有 $K\times P$ 个卷积核和 K 个偏置。

第 l 层的第 p 个特征映射的误差项 $\delta^{(l,p)}$ 如下：

$$\delta^{(l,p)} \triangleq \frac{\partial \mathcal{L}(Y, \hat{Y})}{\partial Z^{(l,p)}} = \frac{\partial X^{(l,p)}}{\partial Z^{(l,p)}} \cdot \frac{\partial \mathcal{L}(Y, \hat{Y})}{\partial X^{(l,p)}}$$

$$= f'(Z^{(l)}) \odot \sum_{k=1}^{K} \mathbf{rot180}(W^{(l+1,k,p)}) \tilde{\otimes} \frac{\partial \mathcal{L}(Y, \hat{Y})}{\partial Z^{(l+1,k)}}$$

$$= f'(Z^{(l)}) \odot \sum_{k=1}^{K} \mathbf{rot180}(W^{(l+1,k,p)}) \tilde{\otimes} \delta^{(l+1,k)}$$

连接表 可以让每一个输出特征映射仅依赖于少数几个输入特征映射（而不是全部）。定义一个连接表（link table）T 来描述输入和输出特征映射之间的连接关系。

如果第 p 个输出特征映射依赖于第 d 个输入特征映射，则 $T_{p,d} = 1$，否则为 0。

$$\boldsymbol{Y}^p = f\left(\sum_{\substack{d, \\ T_{p,d}=1}} \boldsymbol{W}^{p,d} \otimes \boldsymbol{X}^d + b^p \right)$$

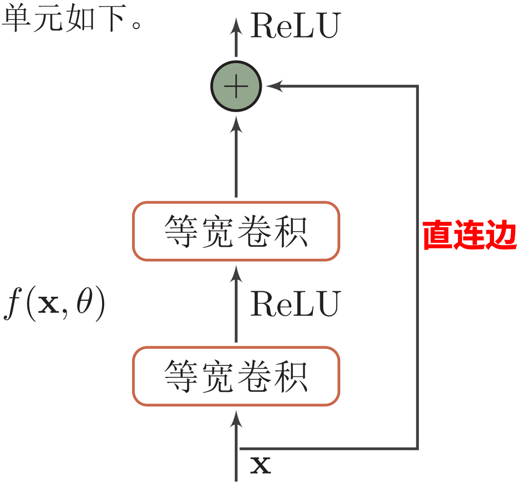
残差网络

残差网络（Residual Network，ResNet）是通过给非线性的卷积层增加直连边的方式来提高信息的传播效率。残差网络的思想并不局限卷积神经网络。

将目标函数拆分成两部分：恒等函数$\mathbf{x}$和残差函数（residue function）$h(\mathbf{x})-\mathbf{x}$两个部分，

$$h(\mathbf{x}) = \underbrace{\mathbf{x}}_{\text{恒等函数}} + \underbrace{\big(h(\mathbf{x}) - \mathbf{x}\big)}_{\text{残差函数}}$$

实际中后者更容易学习 [He et al., 2016]：让非线性单元 $f(\mathbf{x}, \theta)$ 去近似残差函数 $h(\mathbf{x})-\mathbf{x}$，并用 $f(\mathbf{x}, \theta)+\mathbf{x}$ 去逼近 $h(\mathbf{x})$。典型的残差单元如下。



残差网络就是将很多个残差单元串联起来构成的一个非常深的网络。

其它卷积方式

转置卷积

卷积操作一般会降低特征的维数。但在一些任务中，需要将低维特征映射到高维，并且希望通过卷积操作来实现。有一个高维向量为$\mathbf{x} \in \mathbb{R}^d$和一个低维向量为$\mathbf{z} \in \mathbb{R}^p$，$p < d$。如果用变换来实现高维到低维的映射，

$$\mathbf{z} = W\mathbf{x},$$

其中$W \in \mathbb{R}^{p \times d}$为转换矩阵。我们可以很容易地通过转置$W$来实现低维到高维的反向映射，即

$$\mathbf{x} = W^{\mathrm{T}}\mathbf{z}.$$

但并不是逆运算，只是形式上的转置关系。

在全连接网络中，忽略激活函数，前向计算和反向传播就是一种转置关系。前向计算时，第$l+1$层的净输入为$\mathbf{z}^{(l+1)} = W^{(l+1)}\mathbf{z}^{(l)}$，反向传播时，第$l$层的误差项为$\delta^{(l)} = (W^{(l+1)})^{\mathrm{T}}\delta^{(l+1)}$。

5维向量$\mathbf{x}$，经过大小为3的卷积核$\mathbf{w}=[w_1, w_2, w_3]^{\mathrm{T}}$进行卷积，得到3维向量$\mathbf{z}$。卷积操作可以写为

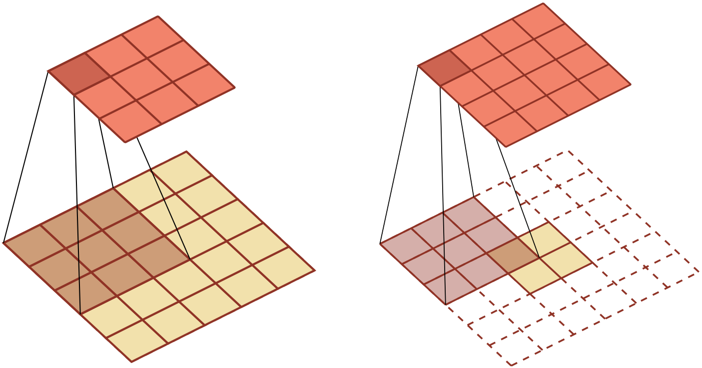
$$\mathbf{z}=\mathbf{w}\otimes\mathbf{x} = \begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix}\cdot\mathbf{x} = C\mathbf{x},$$

3维向量$\mathbf{z}$到5维向量$\mathbf{x}$的映射，可以通过转置来实现。

$$\mathbf{x}=C^{\mathrm{T}}\mathbf{z} = \mathbf{rot180}(\mathbf{w})\tilde{\oplus}\mathbf{z},$$

$\mathbf{z}=\mathbf{w}\otimes\mathbf{x}$和$\mathbf{x}=\mathbf{rot180}(\mathbf{w})\tilde{\oplus}\mathbf{z}$也是形式上的转置关系。因此，将低维特征映射到高维特征的卷积操作称为转置卷积，也称为反卷积。

p维向量$\mathbf{z}$，大小为m的卷积核，希望通过卷积操作来映射到高维向量，只需要对$\mathbf{z}$进行两端补$m-1$个零，然后进行卷积，得到$p+m-1$维的向量。



(a) 卷积，$s=1, p=0$　　(b) 转置卷积，$s=1, p=2$

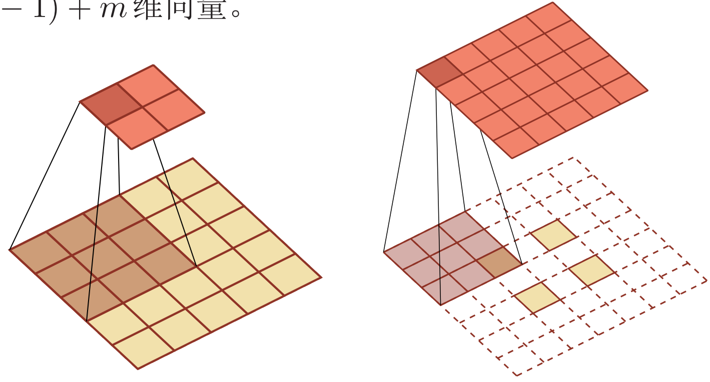
步长$s=1$，无零填充$p=0$的两维卷积和其对应的转置卷积

微步卷积 增加卷积步长 $s > 1$，幅降低特征维数；减少转置卷积的步长 $s < 1$，提高特征维数。

步长 $s < 1$ 的转置卷积也称为微步卷积，可以在输入特征之间插入 0 来间接地使得步长变小。

如果卷积步长为 $s > 1$，希望其对应的转置卷积步长为 $\frac{1}{s}$，需要在输入特征之间插入 $s-1$ 个 0 来使得其移动的速度变慢。

以一维转置卷积为例，对一个 l 维向量 $\mathbf{z}$，和大小为 m 的卷积核，对 $\mathbf{z}$ 两端各补 $p=m-1$ 个零，在每两个向量元素之间插入 $s-1$ 个 0，然后进行步长为 1 的卷积，得到 $1+s\times(l-1)+2(m-1)-m+1=s\times(l-1)+m$ 维向量。



(a) 卷积，$s=2, p=0$ (b) 转置卷积，$l=2, m=3, p=2$

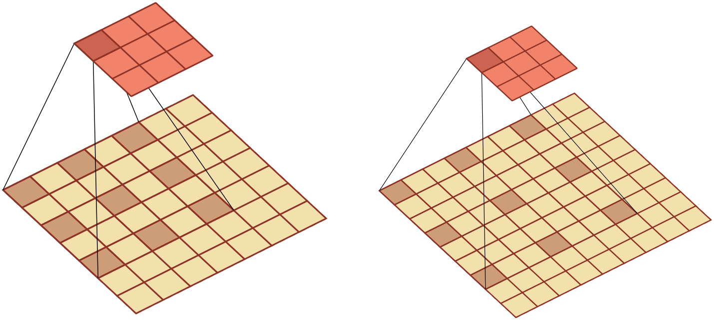
空洞卷积

对于一个卷积层，如果希望增加输出单元的感受野，一般可以通过三种方式实现：（1）增加卷积核的大小；（2）增加层数来实现；（3）在卷积之前进行汇聚操作。前两种操作会增加参数数量，而第三种会丢失一些信息。

空洞卷积也称为膨胀卷积，是一种不增加参数数量，同时增加输出单元感受野的方法。通过给卷积核插入“空洞”来变相地增加其大小。如果在卷积核的每两个元素之间插入d-1个空洞，卷积核的有效大小为

$$m' = m + (m-1) \times (d-1),$$

d称为膨胀率，当$d=1$时卷积核为普通的卷积核。



膨胀率$d=2$　　膨胀率$d=3$

动量因子

$$d_t = \beta d_{t-1} + g(\theta_{t-1})$$

$$\theta_t = \theta_{t-1} - \alpha d_t$$

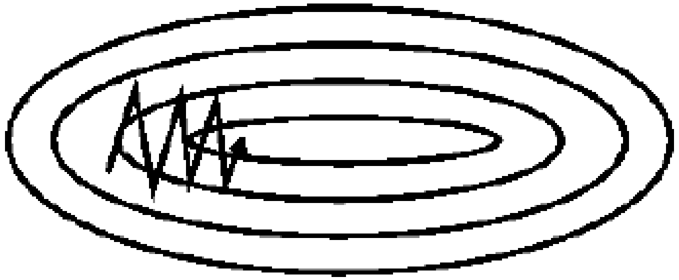
d_t和d_{t-1}：这一次和上一次的更新方向

$g(\theta)$ ：目标函数在θ处的梯度

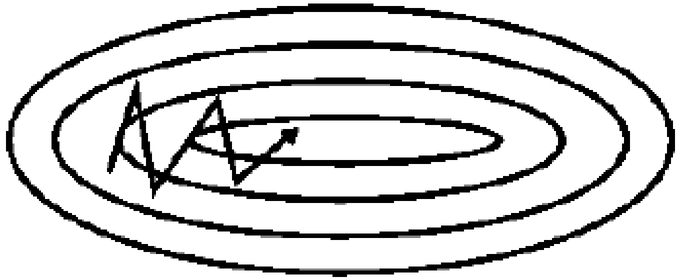
β=0时 ：$d_t = g(\theta_{t-1})$为梯度方向，

$\theta_t = \theta_{t-1} - \alpha g(\theta_{t-1})$，梯度下降！

模拟物体运动时的惯性，即在一定程度上保留之前更新的方向, 使得更新更加平滑，梯度下降的的时候转弯掉头的幅度不那么大了，于是就能够更加平稳、快速地冲向局部最小点。



(a) SGD without momentum



(b) SGD with momentum

动量因子（另一种等价形式）

ρ与η之和不一定为1

$$\Delta\theta_t=\rho\Delta\theta_{t-1}-\eta g(\theta_t)$$

$$\theta_{t+1}=\theta_t+\Delta\theta_t$$

$$\boxed{-\Delta\theta_t}=\boxed{-\rho\Delta\theta_{t-1}}+\eta g(\theta_t)$$

$$\theta_{t+1}=\theta_t-\boxed{(-\Delta\theta_t)}$$

$$\Delta\theta_{-1}=0$$

$$d_t=\beta d_{t-1}+g(\theta_{t-1})$$

$$\theta_t=\theta_{t-1}-\alpha d_t$$

$$d_{t+1}=\beta d_t+g(\theta_t)$$

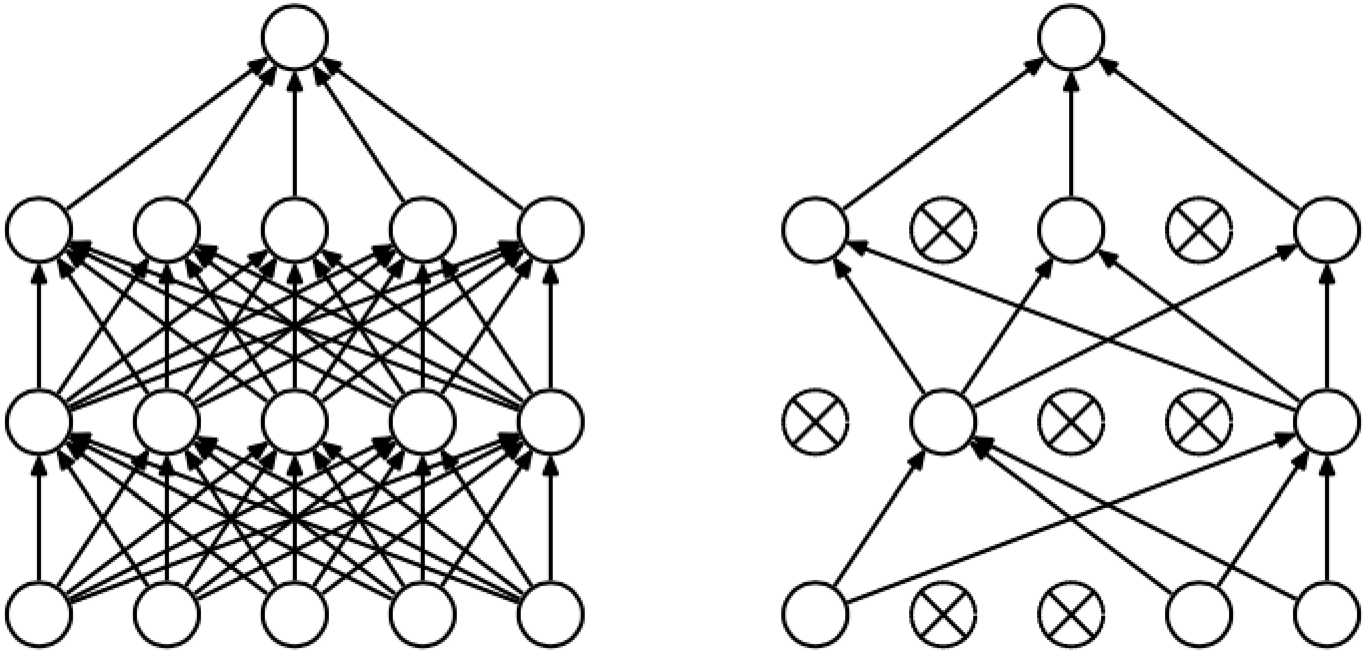
$$\theta_{t+1}=\theta_t-\alpha d_{t+1}$$

$$d_0=0$$

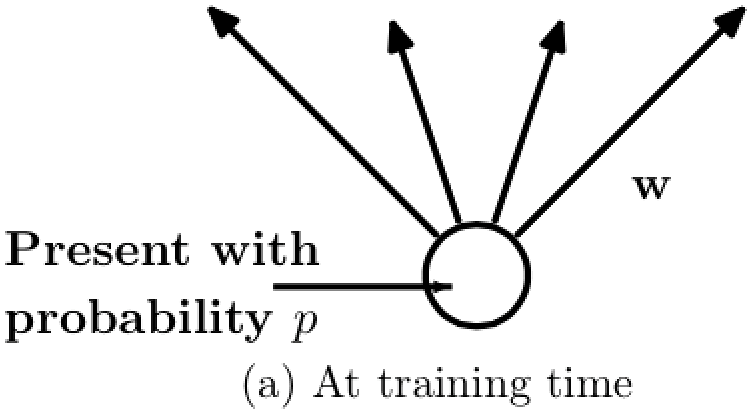
dropout（随机舍弃部分神经元）

- 在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。对于随机梯度下降来说，由于是随机丢弃，故而每一个mini-batch都在训练不同的网络。
- 作用：防止过拟合，在训练阶段阻止神经元的共适应。
- 对于一个有N个节点的神经网络，有了dropout后，就可以看做是2^N个模型的集合了，但此时要训练的参数数目却是不变的。

Dropout(随机舍弃部分神经元)

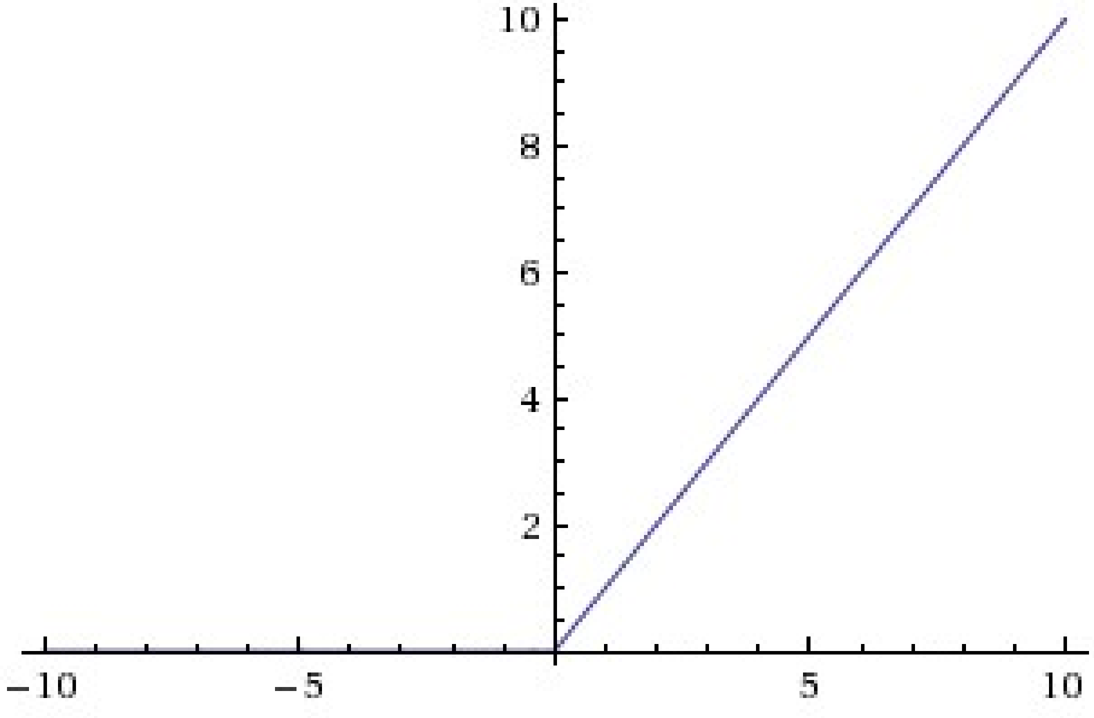


Dropout(随机舍弃部分神经元)



预测的时候，每一个单元的参数要预乘以P。

ReLu(Rectified Linear Units)激励函数



$$\max(0, x)$$

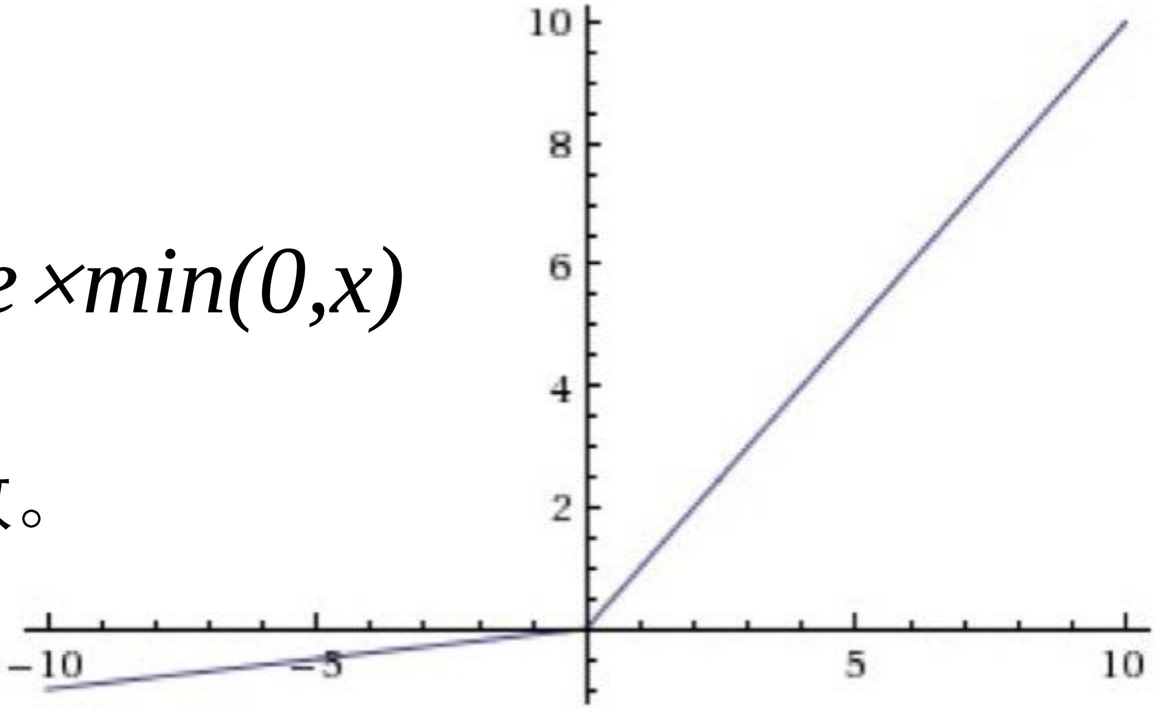
ReLu(Rectified Linear Units)激励函数

优点：对于 SGD 的收敛有巨大的加速作用（Alex Krizhevsky 指出有 6 倍之多）；简单。
缺点：训练时比较脆弱，部分可能“死掉”。合理设置学习速率，会降低这种情况的发生概率。

Leaky ReLU

$$f(x)=max(0,x)+negative\_slope\times min(0,x)$$

*negative_slope*是一个小的非零数。



为解决“ ReLU 死亡”问题的尝试。

ReLU 中当 x<0 时，函数值为 0 。而 Leaky ReLU 则是给出一个很小的梯度值，比如 0.01 。