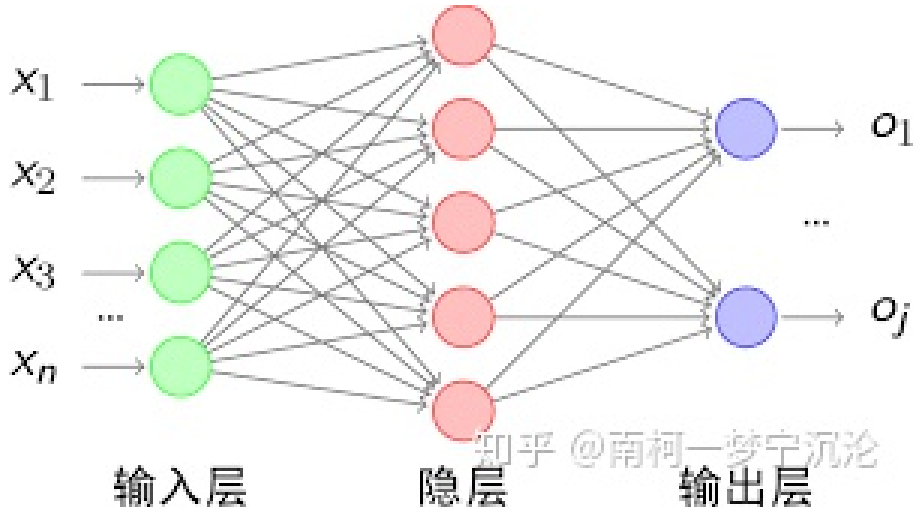
问题

现在才意识到，卷积神经网络在不同的层上的反向传播的计算公式不一样，之前一直按照全连接层的那种简单反向传播去理解了。

全连接层反向传播



在数据表示上，将全连接神经网络的每一层神经元都表示为一个列向量。每一层的神经元，会将上一层神经元的输出作为输入，通过乘上权重矩阵以及加上列向量形式的偏置项。得到激活前的输出值，最后通过激活函数得到该层最终激活后的输出：

$$z^l = W^l * a^{l-1} + b^l \\ a^l = \sigma(z^l) \tag{1}$$

z^l 表示第 l 层 (l = 1, 2, 3, ... , L) 经过激活函数之前的输出，而 a^l 表示第 l 层经过激活函数的输出，σ 表示激活函数。注意，每层的输入以及输出都是一个一维的列向量，我们假设上一层的输出是m×1的列向量，而当前层的输出是n×1的列向量，那么权重矩阵的维度应该是多少呢？应该为n×m。而当前层偏置项的维度为n×1。

如此一来，在我们有一个输入列向量x时，通过一层层的计算，就可以得到我们最终神经网络的输出y。这样神经网络的前向传播就完成了。

而前向传播完成之后，需要更新网络的参数以使得网络实际的输出与正确的输出的差异越来越小，也就是反向传播，我们首先需要定义一个误差函数，这里使用简单直观的**均方误差损失函数**：

$$C = \frac{1}{2}||a^L - y||_2^2 \tag{2}$$

其中的 a^L 经过 L 层全连接后的输出，y 为训练数据中对应输入 x 实际的输出值。求得损失 C 之后，下一步就是利用求得的误差对神经网络中的参数进行更新，即对各层的权重矩阵 W^l 和偏置列向量 b^l 进行更新，使得神经网路的误差减小，达到训练的目的。

在这里我们使用一种叫梯度下降的迭代算法完成参数的更新，通过求出误差对各个参数的梯度大小，令各参数向导数减小的方向变化即可。所以，我们现在的任务是求出误差函数对每个参数的导数。

为了方便进一步的计算推导，以及避免重复计算，我们引入一个中间量 δ^l ，我们称它为**第 l 层的 delta 误差**，表示误差函数对于神经网络第 l 层激活前输出值的偏导数，即 $\delta^l = \frac{\partial C}{\partial z^l}$。

首先，根据神经网络误差函数的定义式，我们可以很容易地求出输出层的delta误差 δ^L ：

$$\delta^L = \frac{\partial C}{\partial z^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L} = (a^L - y) \odot \delta'(z^L) \tag{3}$$

公式中的 $\odot$ 表示 Hardmard 积，即对应逐元素相乘。注意输出层的 delta 误差 δ^L 与损失函数的定义相关，不同的损失函数得到不同的计算结果，在本文中损失函数以均方误差为例讲解。

求得了输出层的delta误差，误差函数C对于输出层参数的导数，即对权重矩阵以及偏置项的导数可通过输出层的delta误差求得如下，这里使用了求导的链式法则

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial z^L}\frac{\partial z^L}{\partial W^L} = \delta^L (a^{L-1})^T \tag{4}$$
$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial z^L}\frac{\partial z^L}{\partial b^L} = \delta^L * 1$$

在这里注意矩阵乘法的求导即乘上系数矩阵对应的转置，左乘还是右乘需要与求导前保持一致，我们通过分析计算公式中各项的矩阵维度可以验证我们公式在维度上是正确的。

得到了最后一层的delta误差，我们接下来需要将delta误差逆向传播，即不断地根据后一层的delta误差求得前一层的delta误差，最终求得每一层的delta误差。其实在这里我们主要利用的是求导的链式法则。假设我们已经求得第l+1层的delta误差，我们可以将第l层的delta误差表示如下

$$\delta^l = \frac{\partial C}{\partial z^l} = \frac{\partial C}{\partial z^{l+1}}\frac{\partial z^{l+1}}{\partial z^l} = \delta^{l+1} * \frac{\partial z^{l+1}}{\partial z^l} \tag{5}$$

又：

$$z^{l+1} = W^{l+!}a^l + b^{l+!} = W^{l+!}\sigma(z^l) + b^{l+!} \tag{6}$$

因此：

$$\delta^l = (W^{l+1})^T \delta^{l+1} \odot \sigma'(z^l) \tag{7}$$

由于我们之前计算出了最后一层的 delta 误差 δ^L ，通过上式，我们依次可以求得 δ^{L-1} ，δ^{L-2} 一直到第二层的 delta 误差 δ^2 。需要注意的是：第一层为我们的输入，并不存在第一层的 delta 误差，因此我们计算到第二层截止。

在求得每一层的 delta 误差后，我们可以很容易地求出误差函数 C 对于每一层参数的梯度，最后可以通过梯度下降法对每一层的参数进行更新。且在一般情况下，我们往往采用随机梯度下降法 (SGD) ，即一次性训练一批数据，先求得这一批数据中的总的误差，最后根据他们的平均误差值对参数进行更新。即：

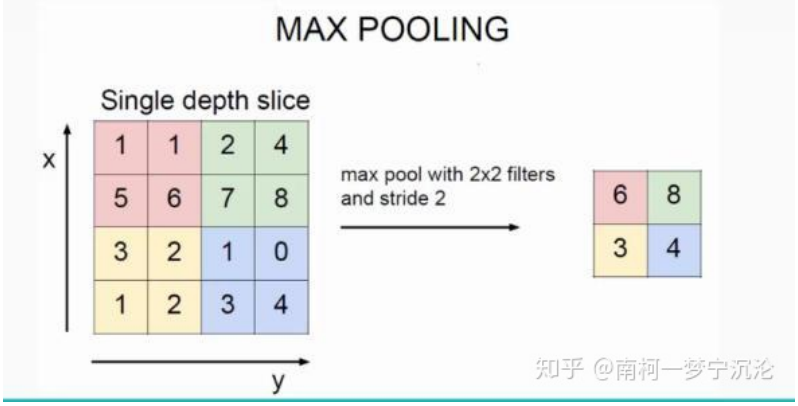
$$W^l = W^l - \eta \frac{\partial(\frac{1}{batch_size} * \sum_{i=1}^{batch_size} C_i)}{\partial W^l} \tag{8}$$

卷积神经网络反向传播推导

分析 delta 误差反向传播过程的简单方法，如果神经网络l+1层某个结点的delta误差要传到l层，我们就去找前向传播时 l+1层的这个结点和第l层的哪些结点有关系，权重是多少，那么反向传播时，delta误差就会乘上相同的权重传播回来。

假设第 l 层有一个结点 a，l+1 层有一个结点 b。两个结点间的连接权重为 w。如果前向传播时，结点 a 对结点 b 的影响是 wa 。而反向传播时，结点 b 的 delta 误差 δ_b 对结点 a 的 delta 误差 δ_a 的影响是 $w\delta_a$ 。它们的系数都为两结点之间的连接权重。

池化层的反向传播



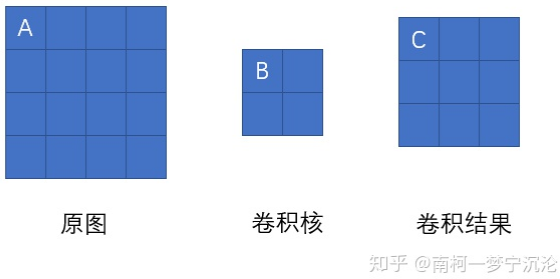
池化层的反向传播比较容易理解，我们以最大池化举例，上图中，池化后的数字6对应于池化前的红色区域，实际上只有红色区域中最大值数字6对池化后的结果有影响，权重为1，而其它的数字对池化后的结果影响都为0。假设池化后数字6的位置delta误差为 δ ，误差反向传播回去时，红色区域中最大值对应的位置delta误差即等于 δ，而其它3个位置对应的delta误差为0。

因此，在卷积神经网络最大池化前向传播时，不仅要记录区域的最大值，同时也要记录下来区域最大值的位置，方便delta误差的反向传播。

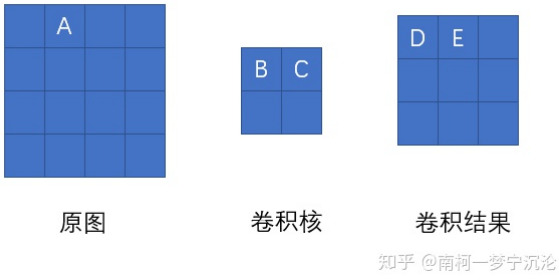
而平均池化就更简单了，由于平均池化时，区域中每个值对池化后结果贡献的权重都为区域大小的倒数，所以delta误差反向传播回来时，在区域每个位置的delta误差都为池化后delta误差除以区域的大小。

卷积层反向传播

虽然卷积神经网络的卷积运算是一个三维张量的图片和一个四维张量的卷积核进行卷积运算，但最核心的计算只涉及二维卷积，因此我们先从二维的卷积运算来进行分析：

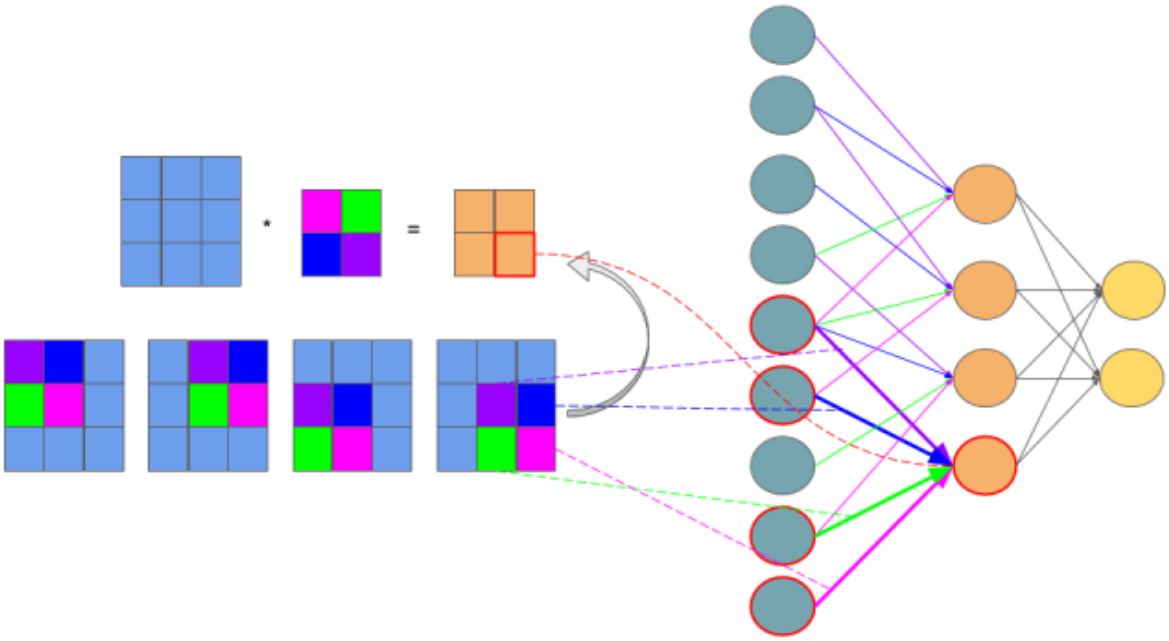


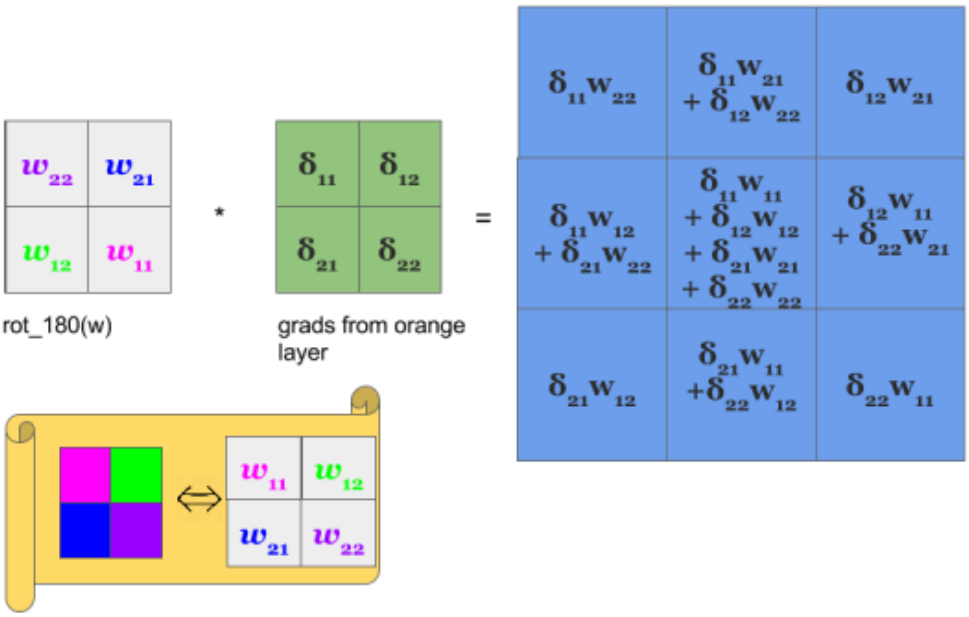
如上图所示，我们求原图A处的delta误差，就先分析，它在前向传播中影响了下一层的哪些结点。显然，它只对结点C有一个权重为B的影响，对卷积结果中的其它结点没有任何影响。因此A的delta误差应该等于C点的delta误差乘上权重B。

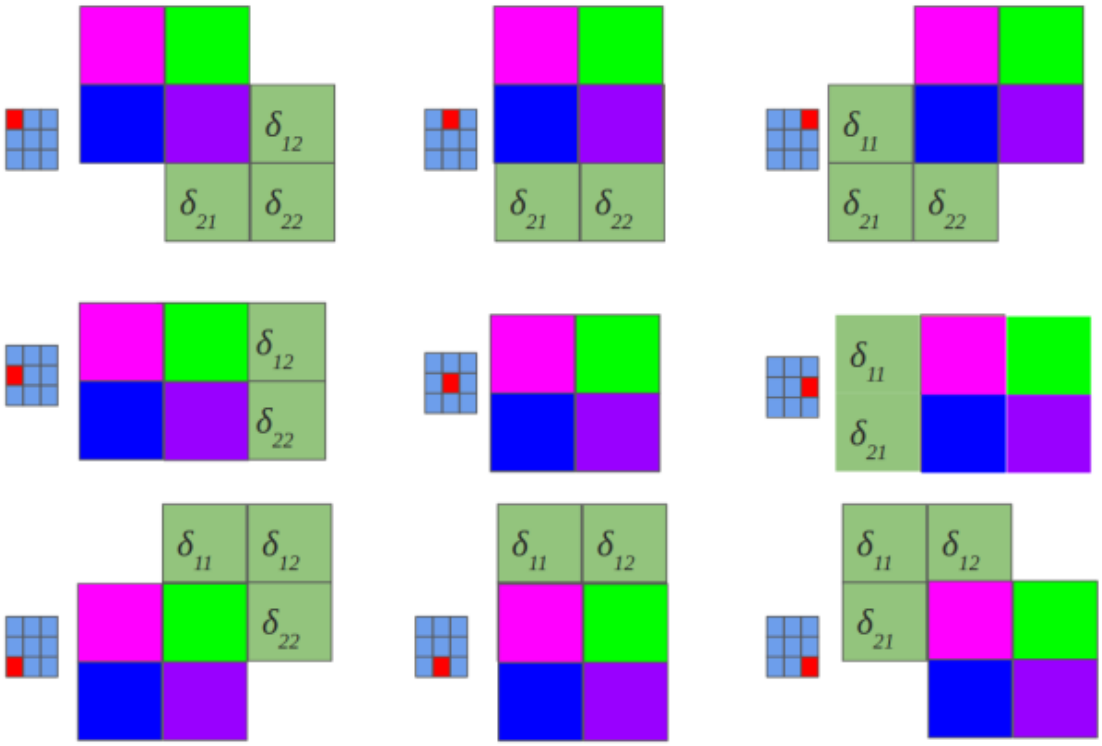


我们现在将原图A点位置移动一下，再看看变换位置后A点的delta误差是多少，同样先分析它前向传播影响了卷积结果的哪些结点。经过分析，A点以权重C影响了卷积结果的D点，以权重B影响了卷积结果的E点。那它的delta误差就等于D点delta误差乘上C加上E点的delta误差乘上B。

大家可以尝试用相同的方法去分析原图中其它结点的delta误差，结果会发现，原图的delta误差，等于卷积结果的delta误差**经过零填充**后，与**卷积核旋转180度**后的卷积。(这一点比较难以理解，具体可以参考参考资料中的第一个链接)







在标准 MLP 中，我们定义第 l 层的第 j 个神经元的 delta 误差为：

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{9}$$

其中 $z_j^l = \sum_k W_{jk}^l a_k^{l-1} + b_j^l, \quad a_j^l = \sigma(z_j^l)$

但在卷积操作中，MLP中矩阵的乘法被卷积所替代，所以我们这里得用 $z_{x,y}^l$ 来替代 z_j^l。

$$z_{x,y}^{l+1} = W^{l+1} * \sigma(z_{x,y}^l) + b_{x,y}^{l+1} = \sum_a \sum_b w_{a,b}^{l+1} \sigma(z_{x-a,y-b}^l) + b_{x,y}^{l+1} \tag{10}$$

接下来，我们将回答这样的一个问题：**为什么在计算CNN的梯度的时候，需要将卷积核旋转180度？**

卷积层中第 l 层的位置为 (x, y) 处的 delta 误差如下：

$$\delta_{x,y}^l = \frac{\partial C}{\partial z_{x,y}^l} = \sum_{x'} \sum_{y'} \frac{\partial C}{\partial z_{x',y'}^{l+1}} \frac{\partial z_{x',y'}^{l+1}}{\partial z_{x,y}^l}$$

在这里，坐标(x',y')是第l+1层中在前向传播中受第l层坐标(x,y)影响到的点，它们不止一个，我们需要将它们加起来。再利用前向传播的关系式可得：

$$\frac{\partial C}{\partial z_{x,y}^l} = \sum_{x'} \sum_{y'} \frac{\partial C}{\partial z_{x',y'}^{l+1}} \frac{\partial z_{x',y'}^{l+1}}{\partial z_{x,y}^l} = \sum_{x'} \sum_{y'} \delta_{x',y'}^{l+1} \frac{\partial (\sum_a \sum_b w_{a,b}^{l+1} \sigma(z_{x'-a,y'-b}^l) + b_{x',y'}^{l+1})}{\partial z_{x,y}^l}$$

以上第一项是根据 delta 误差的概念简化的，而第二项虽然像个怪物，但只要与 $\partial z_{x,y}^l$ 无关的求导之后都为 0，因此上式简化后的结果是：

$$\sum_{x'} \sum_{y'} \delta_{x',y'}^{l+1} \frac{\partial (\sum_a \sum_b w_{a,b}^{l+1} \sigma(z_{x'-a,y'-b}^l) + b_{x',y'}^{l+1})}{\partial z_{x,y}^l} = \sum_{x'} \sum_{y'} \delta_{x',y'}^{l+1} w_{a,b}^{l+1} \sigma'(z_{x,y}^l)$$

上式中的 $x = x' - a$，$y = y' - b$。调换一下位置也就是 $a = x' - x$，$b = y' - y$。因此：

$$\sum_{x'}\sum_{y'}\delta^{l+1}_{x',y'}w^{l+1}_{a,b}\sigma'(z^l_{x,y}) = \sum_{x'}\sum_{y'}\delta^{l+1}_{x',y'}w^{l+1}_{x'-x,y'-y}\sigma'(z^l_{x,y})$$

简化一下书写方式，即：

$$\sum_{x'}\sum_{y'}\delta^{l+1}_{x',y'}w^{l+1}_{x'-x,y'-y}\sigma'(z^l_{x,y}) = \delta^{l+1} * w^{l+1}_{-x,-y}\sigma'(z^l_{x,y})$$

其中 $w^{l+1}_{-x,-y}$ 就是 $w^{l+!}_{x,y}$ 旋转 180 度后的结果：

$$ROT180(w^{l+1}_{x,y}) = w^{l+1}_{-x,-y}$$

所以上面问的问题：**为什么在计算CNN的梯度的时候，需要将卷积核旋转180度？** 的答案很简单，就是根据在二维卷积中对delta误差传播的推导算出来的结果。

最后可以得到根据损失函数对二维卷积中的参数更新的梯度计算方法如下：

$$\frac{\partial C}{\partial w^l_{a,b}} = \sum_x\sum_y\frac{\partial C}{\partial z^l_{x,y}}\frac{\partial z^l_{x,y}}{\partial w^l_{a,b}} = \sum_x\sum_y\delta^l_{x,y}\frac{\partial(\sum_{a'}\sum_{b'}w^l_{a',b'}\sigma(z^l_{x-a',y-b'})+b^l_{x,y})}{\partial w^l_{a,b}} =$$
$$\sum_x\sum_y\delta^l_{x,y}\sigma(z^{l-1}_{x-a,y-b}) = \delta^l_{a,b}*\sigma(z^{l-1}_{-a,-b}) = \delta^l_{a,b}*\sigma(ROT180(z^{l-1}_{a,b}))$$

参考资料

Convolutional Neural Networks backpropagation: from intuition to derivation
全连接神经网络中反向传播算法数学推导
How the backpropagation algorithm works
卷积神经网络(CNN)反向传播算法推导