前面已经推导学习了卷积神经网络之前向传播算法，本篇文章将推导**卷积神经网络之反向传播算法**。在学习卷积神经网络算法之前，希望你对深度神经网络有一定程度的了解，我在之前也有写过相关的文章，包括深度神经网络之前向传播算法、深度神经网络之反向传播算法、深度神经网络之损失函数和激活函数、深度神经网络之正则化，可以先看一下再学习卷积神经网络。

1.DNN反向传播算法

学习CNN(卷积神经网络)反向传播算法之前，我们先回顾下DNN(深度神经网络)反向传播算法。DNN之中，我们是首先计算出输出层的δ^L

$$\delta^L=\frac{\partial J(W,b)}{\partial z^L}=\frac{\partial J(W,b)}{a^L}\odot\sigma'(z^L)$$

然后利用数学归纳法，用δ^{l+1}的值向前求出第l层的δ^l，表达式为

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

有了δ^l表达式，便能够求出W,b的梯度表达式

$$\frac{\partial J(W,b)}{\partial W^l}=\frac{\partial J(W,b,x,y)}{\partial z^l}\frac{\partial z^l}{\partial W^l}=\delta^l(a^{l-1})^T$$

$$\frac{\partial J(W,b)}{\partial b^l}=\frac{\partial J(W,b,x,y)}{\partial z^l}\frac{\partial z^l}{\partial b^l}=\delta^l$$

有了W,b梯度表达式，便可利用梯度下降法来优化W,b，最终求出所有的W,b。了解DNN深度神经网络反向传播算法之后，下面来看下卷积神经网络算法如何求解W,b。

2.CNN反向传播算法

对比深度神经网络反向传播算法，卷积神经网络反向传播算法需要解决以下几个问题。

- 池化层没有激活函数，因此令池化层的激活函数为$\sigma(z)=z$，即激活后便是本身，这样池化层激活函数的导数为1。另池化层在前向传播算法之中，对输入进行了压缩，那么现在反向推倒δ^{l-1}，如何求解呢。
- 卷积层是通过张量进行卷积，而DNN的全连接层是直接进行矩阵乘法计算，然后得到当前层的输出，那么卷积层进行反向传播时如何推导δ^{l-1}呢。
- 对于卷积层，由于和W的运算是卷积，那么从δ^l推导出当前层卷积的W,b方式也不同，针对卷积神经网络如何求解W,b呢。

由于卷积层可以有多个卷积核，各个卷积核之间的处理方式是完全相同的，为了简化算法公式的复杂度，下面推导时只针对卷积层中若干卷积核中的一个。

3.已知池化层δ^l，推导上一隐藏层δ^{l-1}

针对上述问题1，CNN前向传播算法时，池化层一般会采用Max或Average对输入进行池化，池化的区域大小已知。现在我们反过来，从缩小后的误差δ^l，还原前一次较大区域对应的误差。

反向传播时，首先会把δ^l的所有子矩阵大小还原到池化之前的大小。如果是Max方法池化，则把δ^l的所有子矩阵的各个池化区域的值，放在之前做前向传播算法得到最大值的位置。如果是Average方法池化，则把δ^l的所有子矩阵的各个池化区域的值，取平均后放在还原后的子矩阵位置，上述过程一般叫做upsample。

下面我们通过一个简单例子来表示upsample，假设我们池化后的区域大小是2*2，δ^l的第k个子矩阵为

$$\delta_k^l = \begin{bmatrix} 2 & 8 \\ 4 & 6 \end{bmatrix}$$

由于池化区域为2*2，首先将δ_k^l进行还原

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 8 & 0 \\ 0 & 4 & 6 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

如果是Max方法，假设之前在前向传播算法记录的最大值位置分别是左上、右下、右上、左下，则转换后的矩阵为

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 6 & 0 \end{bmatrix}$$

如果是Average方法，则进行平均化，转化后的矩阵为

$$\begin{bmatrix} 0.5 & 0.5 & 2 & 2 \\ 0.5 & 0.5 & 2 & 2 \\ 1 & 1 & 1.5 & 1.5 \\ 1 & 1 & 1.5 & 1.5 \end{bmatrix}$$

至此我们便能够得到上一层$\frac{\partial J(W,b)}{\partial a_k^{l-1}}$，通过下式便能得到$\delta_k^{l-1}$，其中upsample函数完成池化误差矩阵放大于误差重新分配的逻辑

$$\delta_k^{l-1} = \frac{\partial J(W,b)}{\partial a_k^{l-1}} \frac{\partial a_k^{l-1}}{\partial z_k^{l-1}} = upsample(\delta_k^l) \odot \sigma'(z_k^{l-1})$$

最终对于张量δ^{l-1}，我们有

$$\delta^{l-1} = upsample(\delta^l) \odot \sigma'(z^{l-1})$$

3.已知卷积层δ^l，推导上一隐藏层δ^{l-1}

DNN之中，我们已经知道δ^{l-1}和δ^l之间的递推关系如下所示

$$\delta^{l-1} = \frac{\partial J(W,b)}{\partial z^{l-1}} = \frac{\partial J(W,b)}{\partial z^l} \frac{\partial z^l}{\partial z^{l-1}} = \delta^l \frac{\partial z^l}{\partial z^{l-1}}$$

如果想要计算δ^{l-1}和δ^l之间的关系，必须求出$\frac{\partial z^l}{\partial z^{l-1}}$的梯度表达式，注意到$z^l$和$z^{l-1}$的关系为

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

因此能够得到

$$\delta^{l-1} = \delta^l \frac{\partial z^l}{\partial z^{l-1}} = \delta^l rot180(W^l) \odot \sigma'(z^{l-1})$$

上述方程式和DNN之中的类似，区别在于对含有卷积的式子进行求导时，卷积核需要旋转了180度。翻转180度的意思是上下翻转一次，接着左右翻转一次。在DNN中只是对矩阵W进行转置，那么这里为什么需要进行翻转呢？下面通过一个简单例子来描述为什么求导后卷积核需要翻转。

假设我们第$l-1$层的输出a^{l-1}是个3*3的矩阵，第l层的卷积核W^l是一个2*2的矩阵，卷积过程采用1像素的步幅，那么输出是一个2*2的矩阵，为简化计算，假设b^l都是0，则有

$$a^{l-1} * W^l = z^l$$

列出a, W, z的矩阵表达式为

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}$$

利用卷积的定义，我们能够得到

$$z_{11} = a_{11}w_{11} + a_{12}w_{12} + a_{21}w_{21} + a_{22}w_{22}$$

$$z_{12} = a_{12}w_{11} + a_{13}w_{12} + a_{22}w_{21} + a_{23}w_{22}$$

$$z_{21} = a_{21}w_{11} + a_{22}w_{12} + a_{31}w_{21} + a_{32}w_{22}$$

$$z_{22} = a_{22}w_{11} + a_{23}w_{12} + a_{32}w_{21} + a_{33}w_{22}$$

接着进行求导

$$\nabla a^{l-1} = \frac{\partial J(W,b)}{\partial a^{l-1}} = \frac{\partial J(W,b)}{\partial z^l} \frac{\partial z^l}{\partial a^{l-1}} = \delta^l \frac{\partial z^l}{\partial a^{l-1}}$$

从上式可以看出，对于a^{l-1}的梯度误差∇a^{l-1}，等于第l层的梯度误差乘以$\frac{\partial z^l}{\partial a^{l-1}}$，而$\frac{\partial z^l}{\partial a^{l-1}}$对应上面例子之中相关联的w值。假如我们的$z$矩阵对应的反向传播误差是$\delta_{11}, \delta_{12}, \delta_{21}, \delta_{22}$组成的2*2矩阵，则利用上面梯度的式子和四个等式，我们可以分别写出∇a^{l-1}的9个标量的梯度。

比如对于a_{11}的梯度，在上面四个等式之中，a_{11}只和z_{11}有乘积关系，因此有

$$\nabla a_{11} = \delta_{11}w_{11}$$

同样我们能够得到

$$\nabla a_{12} = \delta_{11}w_{12} + \delta_{12}w_{11}$$

$$\nabla a_{13} = \delta_{12}w_{12}$$

$$\nabla a_{21} = \delta_{11}w_{21} + \delta_{21}w_{11}$$

$$\nabla a_{22} = \delta_{11}w_{22} + \delta_{12}w_{21} + \delta_{21}w_{12} + \delta_{22}w_{11}$$

$$\nabla a_{23} = \delta_{12} w_{22} + \delta_{22} w_{12}$$

$$\nabla a_{31} = \delta_{21} w_{21}$$

$$\nabla a_{32} = \delta_{21} w_{22} + \delta_{22} w_{21}$$

$$\nabla a_{33} = \delta_{22} w_{22}$$

针对上面的9个等式，我们可以采用矩阵卷积的形式表示。为了符合梯度计算，在误差矩阵周围填充一圈0，此时我们将卷积核翻转后和反向传播的梯度误差进行卷积，便能够得到前一次的梯度误差。通过例子也便能够了解为什么卷积核需要翻转180度。

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_{11} & \delta_{12} & 0 \\ 0 & \delta_{12} & \delta_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix} = \begin{bmatrix} \nabla_{11} & \nabla_{12} & \nabla_{13} \\ \nabla_{21} & \nabla_{22} & \nabla_{23} \\ \nabla_{31} & \nabla_{32} & \nabla_{33} \end{bmatrix}$$

4.已知卷积层δ^l，推导该层W, b梯度

根据上面的方法，我们已经递推得到每一层的梯度误差δ^l，那么如何求解卷积神经网络的W, b呢。对于全连接层，可以按照DNN的反向传播算法求出该层的W, b的梯度。而池化层没有W, b，因此只要求解卷积层的W, b。

注意到卷积层z和W, b的关系为

$$z^l = a^{l-1} * W^l + b$$

因此能够得到

$$\frac{\partial J(W,b)}{\partial W^l} = \frac{\partial J(W,b)}{\partial z^l} \frac{\partial z^l}{\partial W^l} = \delta^l * a^{l-1}$$

注意到此时的卷积核并没有进行翻转，因为此时进行的是层内的求导，而不是反向传播到上一层的求导。下面我们通过一个例子分析为什么没有进行翻转，这里输入的是矩阵，而不是张量，对于第l层，某个卷积核矩阵W的导数可以表示如下

$$\frac{\partial J(W,b)}{\partial W^l_{pq}} = \sum_i \sum_j (\delta^l_{ij} x^{l-1}_{i+p-1, j+q-1})$$

假如输入的a是4*4的矩阵，卷积核W是3*3的矩阵，输出z是2*2的矩阵，那么反向传播误差z的梯度误差δ也是2*2的矩阵。根据上面的式子，能够得到

$$\frac{\partial J(W,b)}{\partial W^l_{11}} = a_{11}\delta_{11} + a_{12}\delta_{12} + a_{21}\delta_{21} + a_{22}\delta_{22}$$

$$\frac{\partial J(W,b)}{\partial W^l_{12}} = a_{12}\delta_{11} + a_{13}\delta_{12} + a_{22}\delta_{21} + a_{23}\delta_{22}$$

$$\frac{\partial J(W,b)}{\partial W^l_{13}} = a_{13}\delta_{11} + a_{14}\delta_{12} + a_{23}\delta_{21} + a_{24}\delta_{22}$$

$$\cdots$$

最终能够得到9个式子，整理成矩阵可得如下公式，据此我们能够看出为什么没有进行翻转。

$$\frac{\partial J(W,b)}{\partial W^l}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} * \begin{bmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{bmatrix}$$

对于b则有点特殊，因为δ^l是三维张量，而b只是一个向量，不像DNN那样直接和δ^l相等。通常的做法是将δ^l的各个子矩阵的项分别求和，得到误差向量，最后可得b的梯度为

$$\frac{\partial J(W,b)}{\partial b^l}=\sum_{u,v}(\delta^l)_{u,v}$$

5.CNN反向传播算法总结

输入：m个图片样本，CNN模型的层数L和所有隐藏层的类型。对于卷积层，要定义卷积核的大小K，卷积核子矩阵的维度F，填充大小P，步幅S。对于池化层，要定义池化层区域大小k和池化标准(Max或Average)。对于全连接层，定义全连接层的激活函数(输出层除外)和各层神经元的个数。梯度迭代步长α，最大迭代次数Max和停止迭代阈值ϵ。

输出：CNN模型各隐藏层与输出层的W,b。

- 初始化各隐藏层和输出层的W,b值为随机值。
- for iter from 1 to Max
  - for i=1 to m
    - 将CNN输入a^1设置为x_i对应的张量。
    - for l=2 to L-1，前向传播算法
      - 如果当前层是全连接层，则$a^{i,l}=\sigma(z^{i,l})=\sigma(W^l a^{i,l-1}+b^l)$。
      - 如果当前层是卷积层，则$a^{i,l}=\sigma(z^{i,l})=\sigma(W^l * a^{i,l-1}+b^l)$。
      - 如果当前层是池化层，则$a^{i,l}=pool(a^{i,l-1})$。这里的pool指的是按照池化区域大小k和池化标准将输入张量缩小的过程。
    - 对于输出层$a^{i,L}=Softmax(z^{i,L})=Softmax(W^L * a^{i,L-1}+b^L)$。
    - 通过损失函数计算输出层$\delta^{i,L}$。
    - for l=L-1 to 2，后向传播算法
      - 如果当前层是全连接层，则$\delta^{i,l}=(W^{l+1})^T\delta^{i,l+1}\odot\sigma'(z^{i,l})$。
      - 如果当前层是卷积层，则$\delta^{i,l}=\delta^{i,l+1}*rot180(W^{l+1})\odot\sigma'(z^{i,l})$。
      - 如果当前是池化层，则$\delta^{i,l}=upsample(\delta^{i,l+1})\odot\sigma'(z^{i,l})$。
  - for l=2 to L，更新第l层的W^l,b^l
    - 如果当前层是全连接层，则$W^l=W^l-\alpha\sum_{i=1}^m\delta^{i,l}(a^{i,l-1})^T$，$b^l=b^l-\alpha\sum_{i=1}^m\sum_{u,v}(\delta^{i,l})_{uv}$。
    - 如果当前层是卷积层，则对于每个卷积核有$W^l=W^l-\alpha\sum_{i=1}^m\delta^{i,l}*rot180(a^{i,l-1})$，$b^l=b^l-\alpha\sum_{i=1}^m\sum_{u,v}(\delta^{i,l})_{uv}$。
  - 如果所有的W,b的变化值都小于停止迭代阈值ϵ，则跳出迭代循环。
- 输出各隐藏层与输出层的线形关系系数矩阵W和偏倚变量b。

6.推广

更多内容请关注公众号**谓之小一**，若有疑问可在公众号后台提问，随时回答，欢迎关注，内容转载请注明出处。

「谓之小一」希望提供给读者别处看不到的内容，关于互联网、数据挖掘、机器学习、书籍、生活……

- 知乎：@谓之小一
- 公众号：@谓之小一
- GitHub：@weizhixiaoyi
- 技术博客：https://weizhixiaoyi.com

谓之小一

长按关注微信公众号