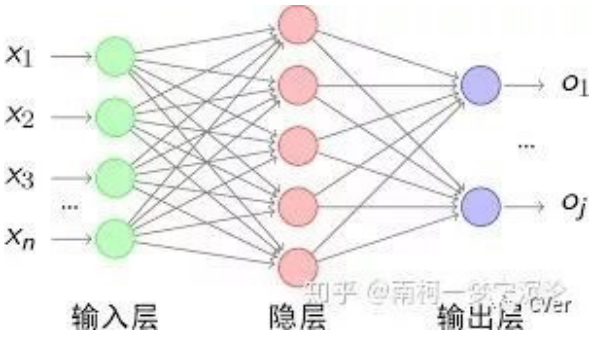
全连接神经网络中反向传播算法数学推导

作者：南柯一梦宁沉沦

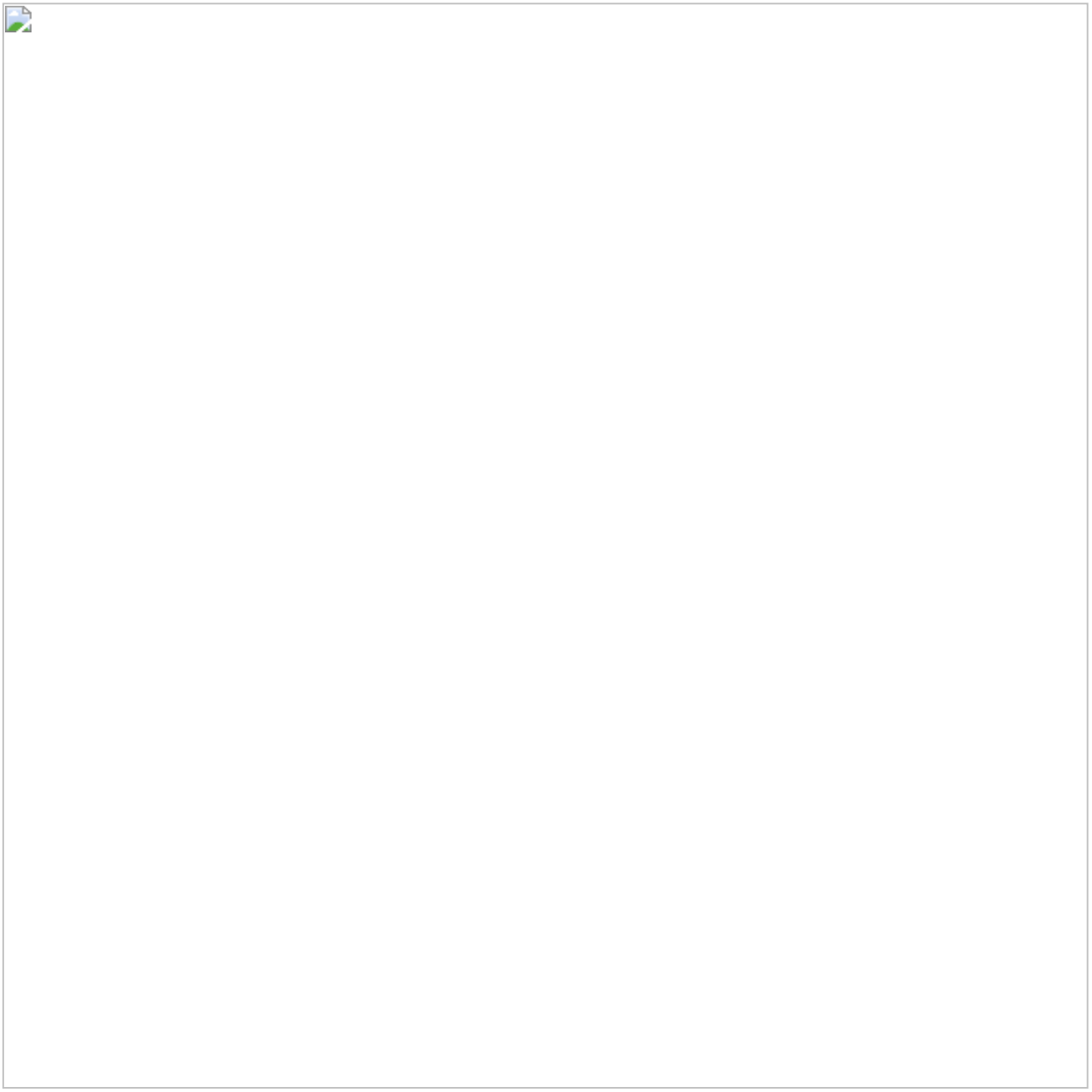
https://zhuanlan.zhihu.com/p/61863634

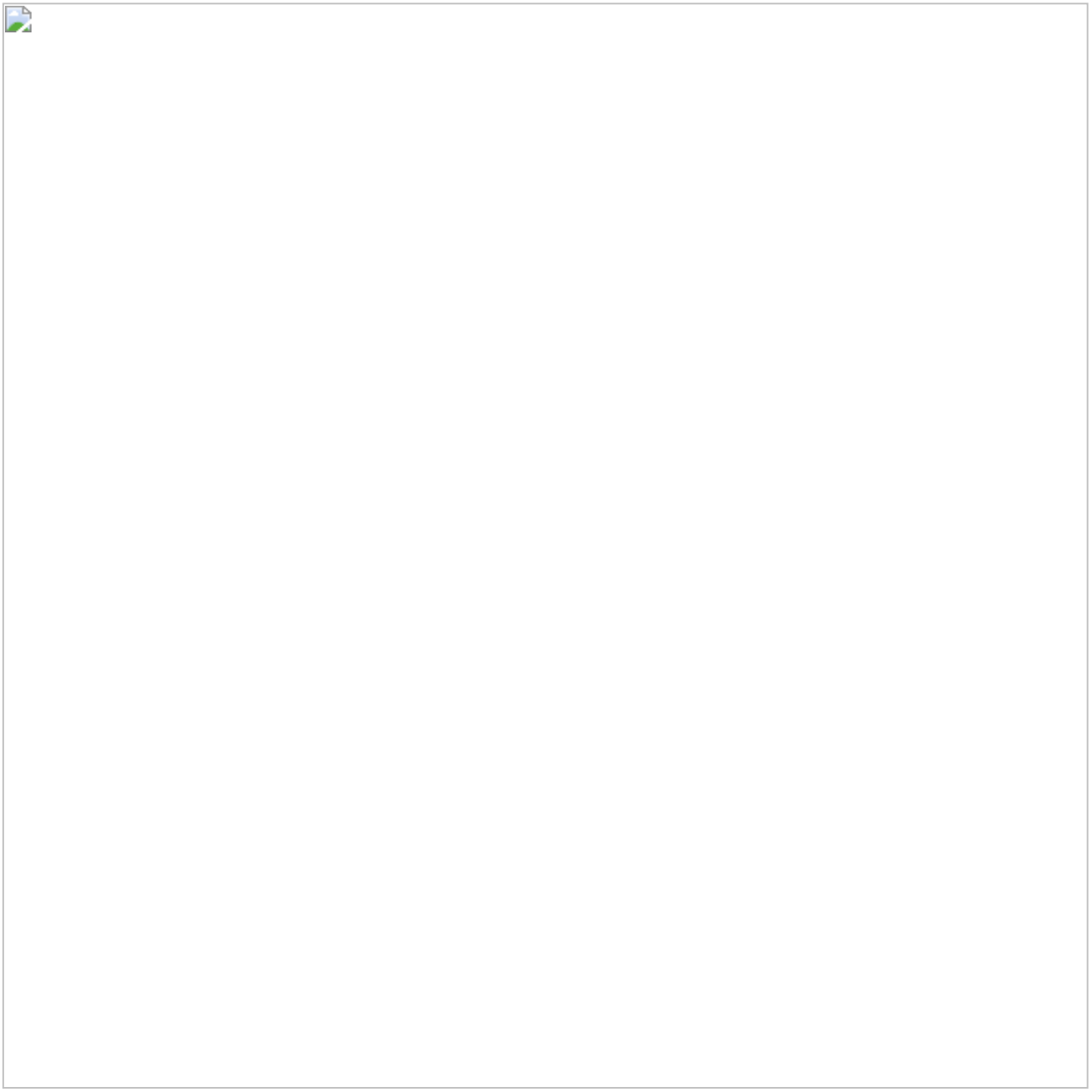
本文以神经网络中最简单的全连接神经网络为例介绍反向传播算法的理论推导。因为它涉及到的数学理论相对其它网络更为简单，仅需要对矩阵的乘法运算以及微积分中求导链式法则有较好的理解。全连接神经网络中的反向传播算法是理解其它更加复杂的神经网络如卷积神经网络反向传播算法的重要基础。



三层全连接神经网络

我们将感知机中每一层都表示为一个列向量。每一层的感知机，会将上一层感知机的输出作为输入，通过乘上权重矩阵以及加上列向量形式的偏置项，即可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，具体计算公式如下：





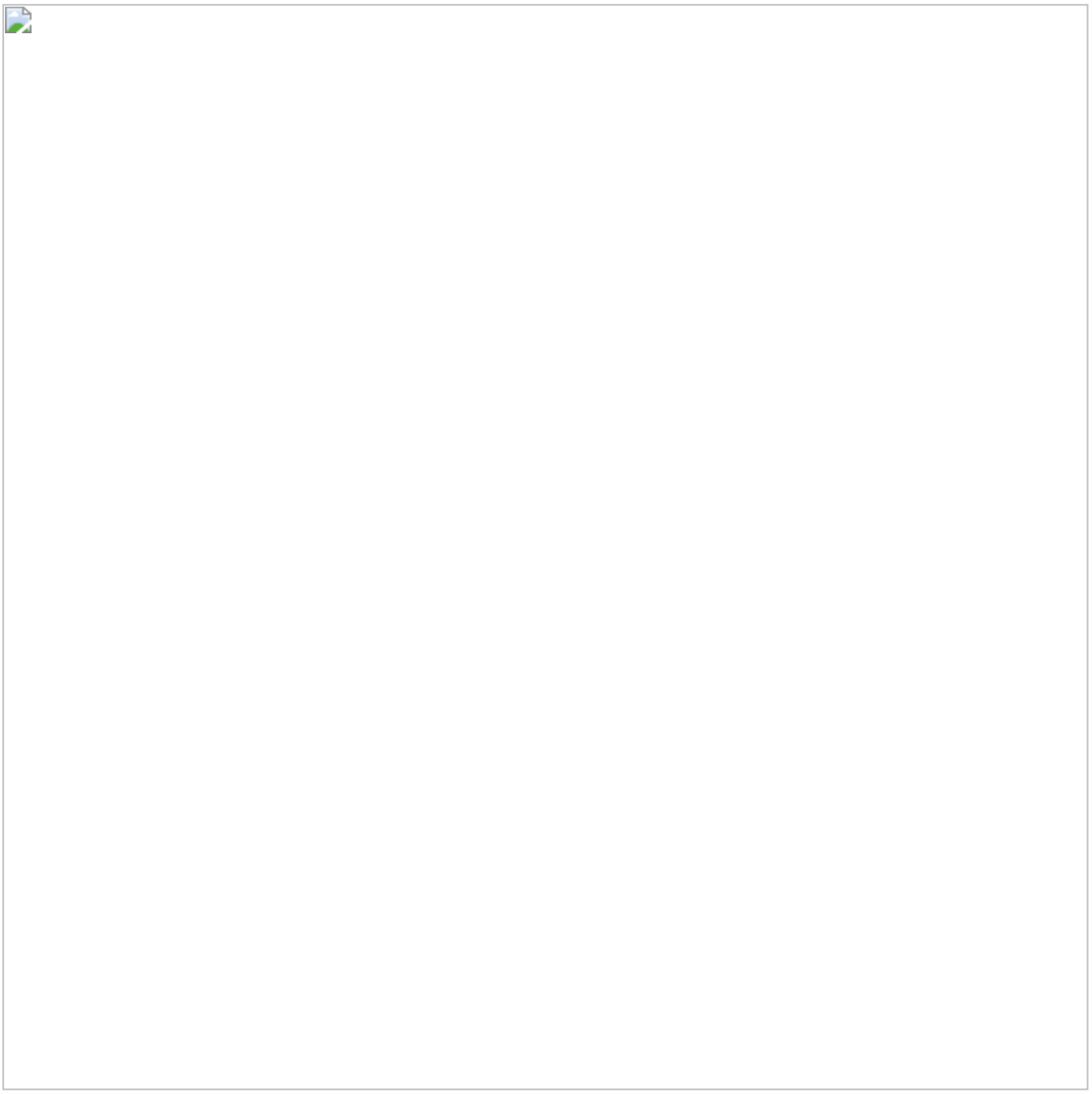

表示第l层(l=1,2,...,L)经过激活函数之前的输出，而

表示第l层经过激活函数之后的输出，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。这样神经网络的前向传播就完成了。

而我们的目标是得到一个神经网络，让它对我们的输入，能给出正确的输出。为了达到这个目的，我们需要预先给神经网络喂入大量标注过的数据，即不仅给它输入数据，也

告诉它什么是对应正确的输出，让神经网络自己去学习调整内部的参数。
在训练过程中，我们首先需要定义一个误差函数（也称损失函数），用来度量神经网络的输出与正确的输出之间的差异。为了便于理解，我们这里使用简单直观的均方误差损失函数：



我们用L代表多层感知机总的层数，

表示多层感知机第L层经过激活函数后的输出，也即神经网络所预测的输出值。而y是训练数据中对应输入x实际的输出值y。
经过前向传播之后，我们就得到了当前神经网络的误差大小，下一步则是利用求得的误差对神经网络的参数进行更新，即对各层的权重矩阵

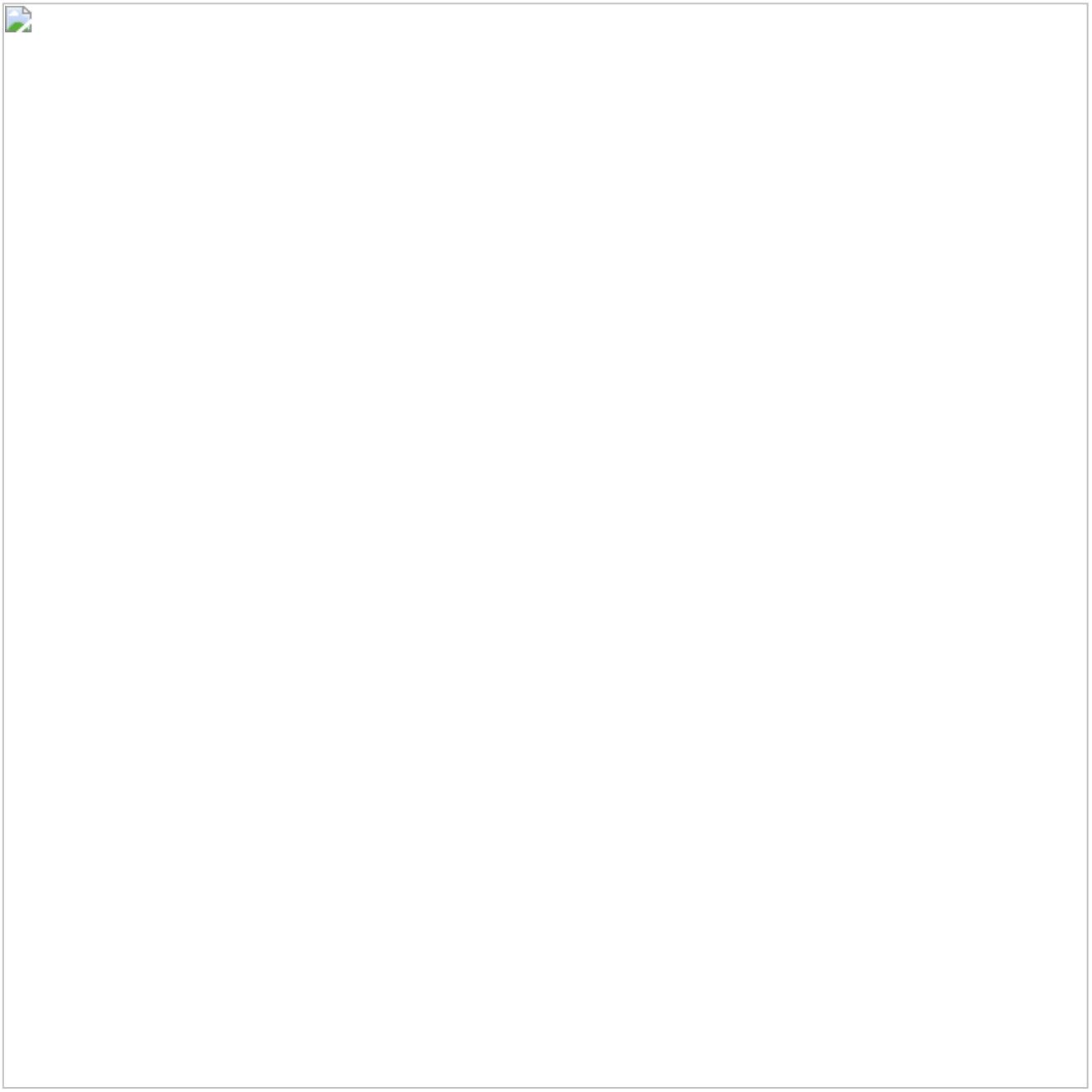
和偏置项

进行更新，使神经网络的误差减小，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导数大小，令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

，我们称它为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

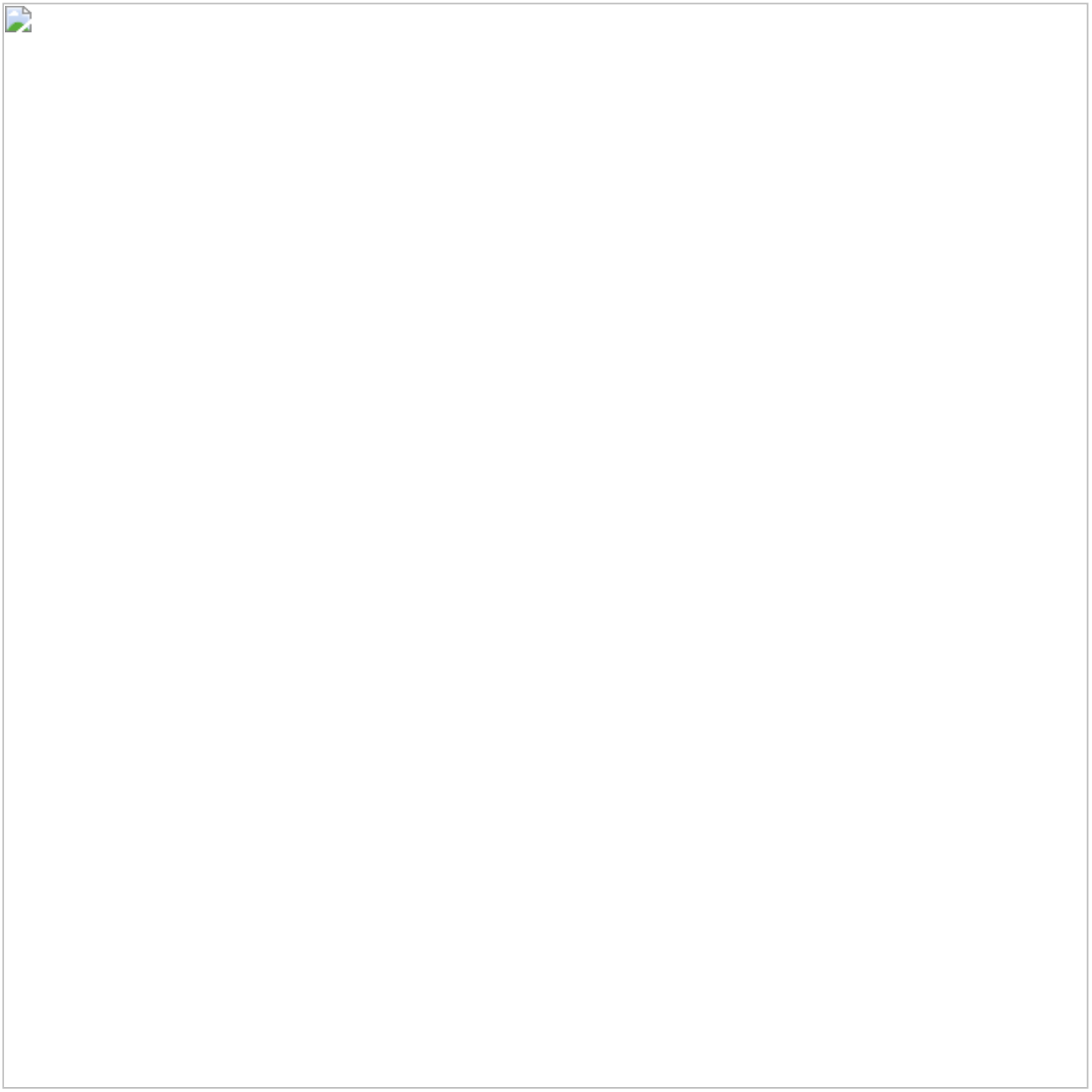
根据神经网络误差函数的定义式，我们可以很容易地求出输出层的delta误差

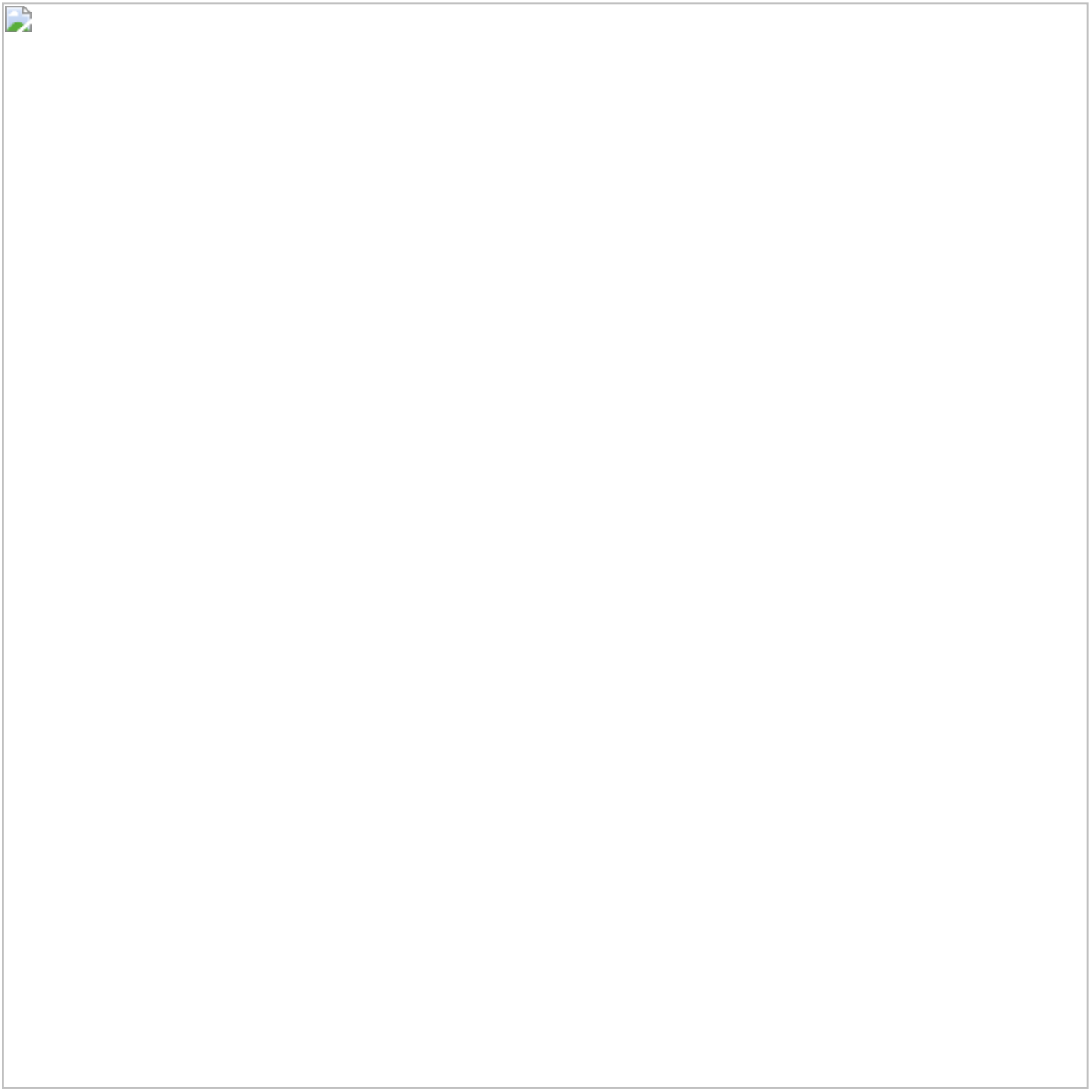


在公式里，

表示Hadmard积，即对应逐元素相乘，与矩阵乘法相区分。

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相应求得为

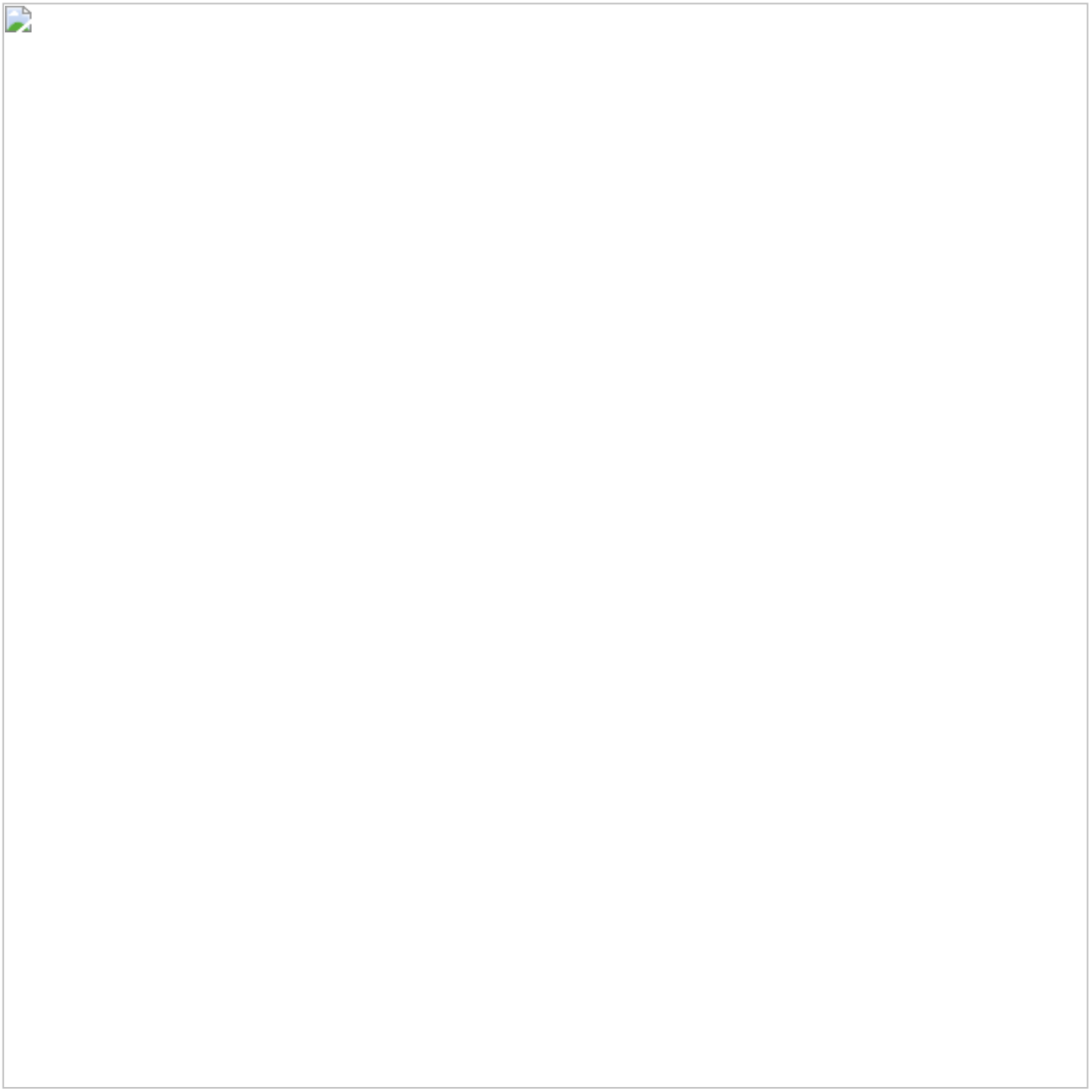




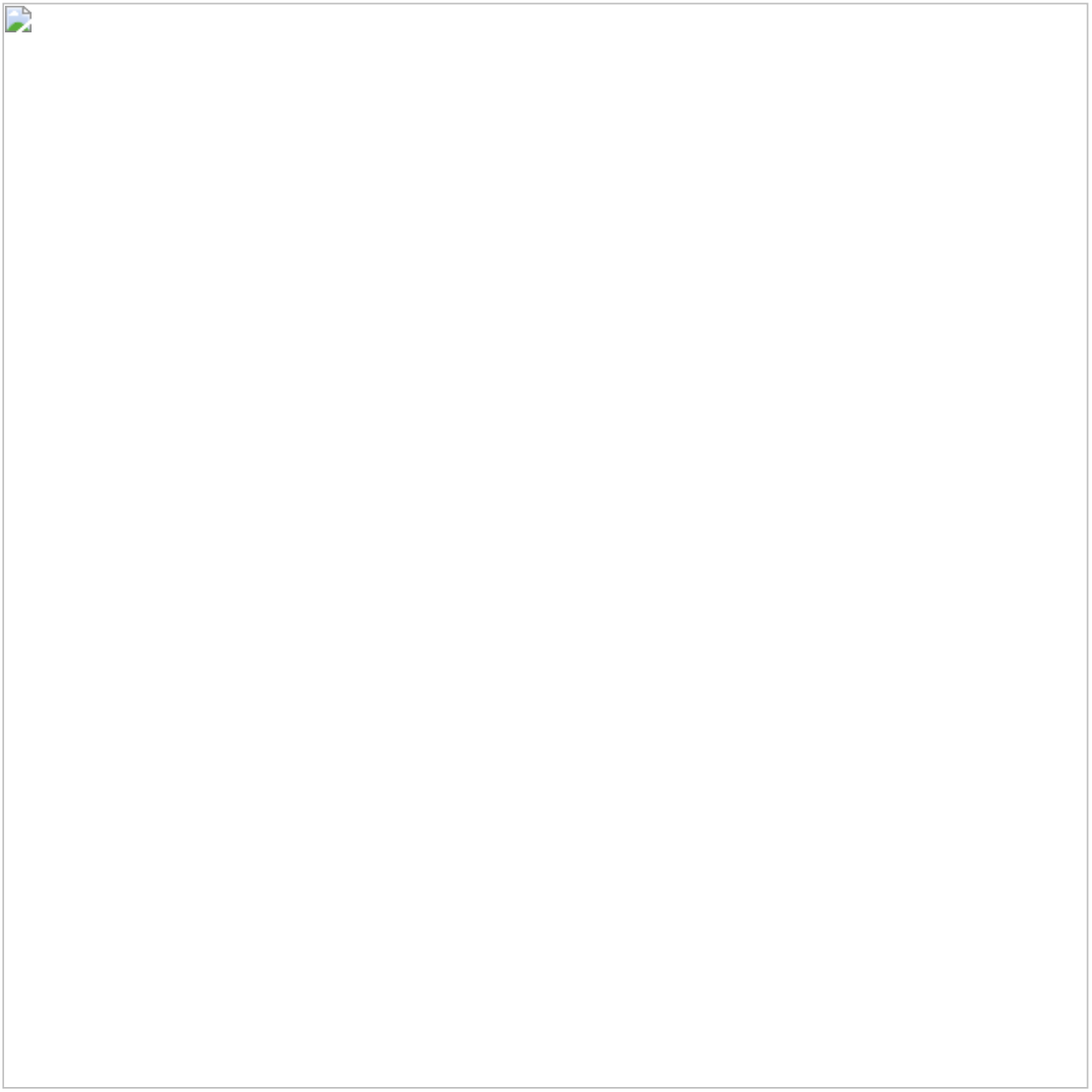
在这里注意矩阵乘法的求导即乘上系数矩阵对应的转置，左乘还是右乘需要与求导前保持一致，我们通过分析计算公式中各项的矩阵维度可以验证我们公式在维度上是正确的。

在这里我们可以很容易看到，一旦求出了当前层的delta误差，误差函数对当前层各参数的导数便可以相应的求出。

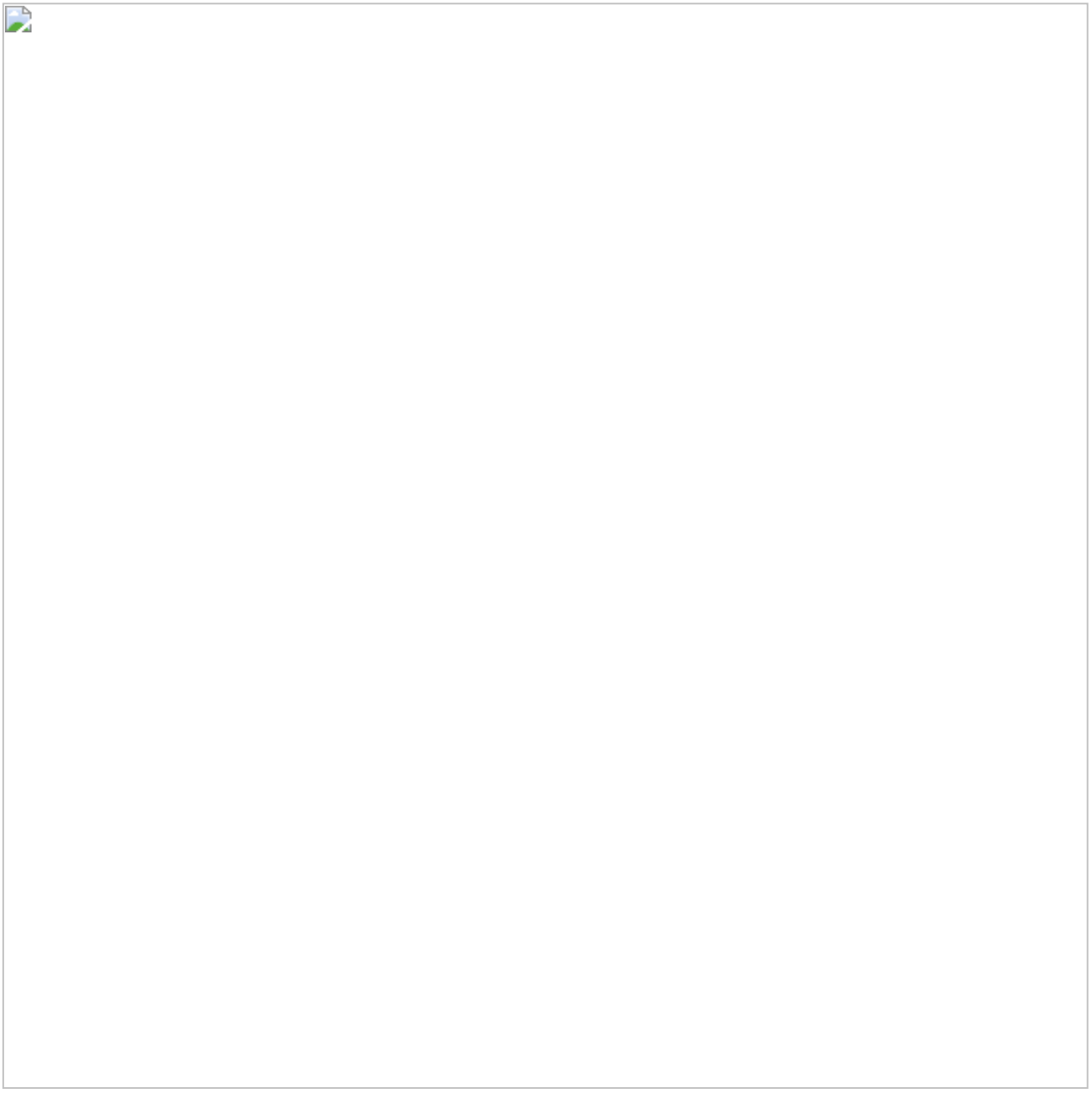
得到了最后一层的delta误差，我们接下来利用我们的主角——反向传播算法，将delta误差逆向传播，即不断地根据后一层的delta误差求得前一层的delta误差，最终求得每一层的delta误差。其实在这里我们主要利用的是求导的链式法则。假设我们已经求得第l+1层的delta误差，我们可以将第l层的delta误差表示如下。



又：



因此：



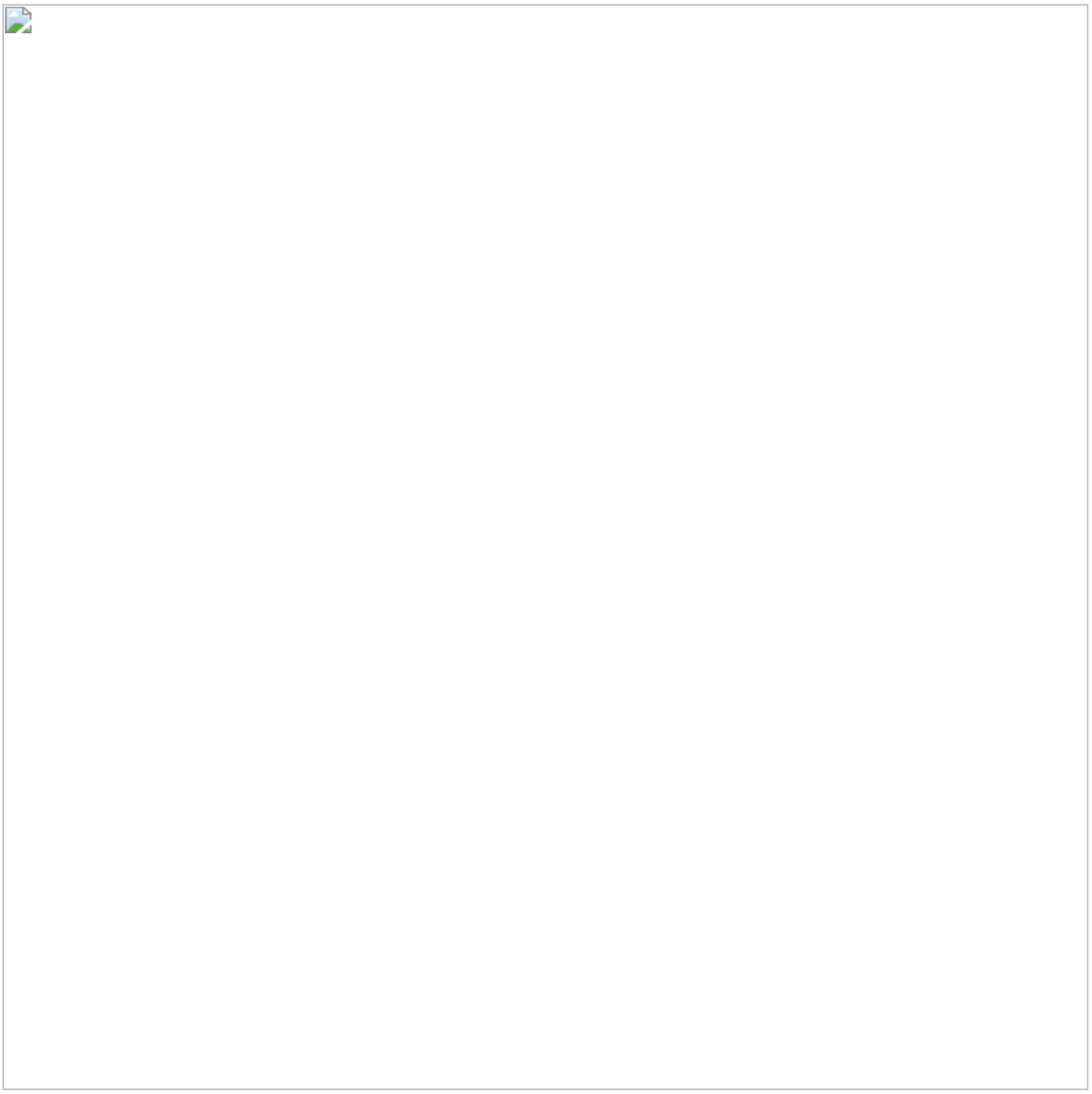
在这里我们需注意的是求导后矩阵运算是左乘还是右乘需要与求导前保持一致，并且需要经过转置。同样的我们可以通过分析维度来验证。

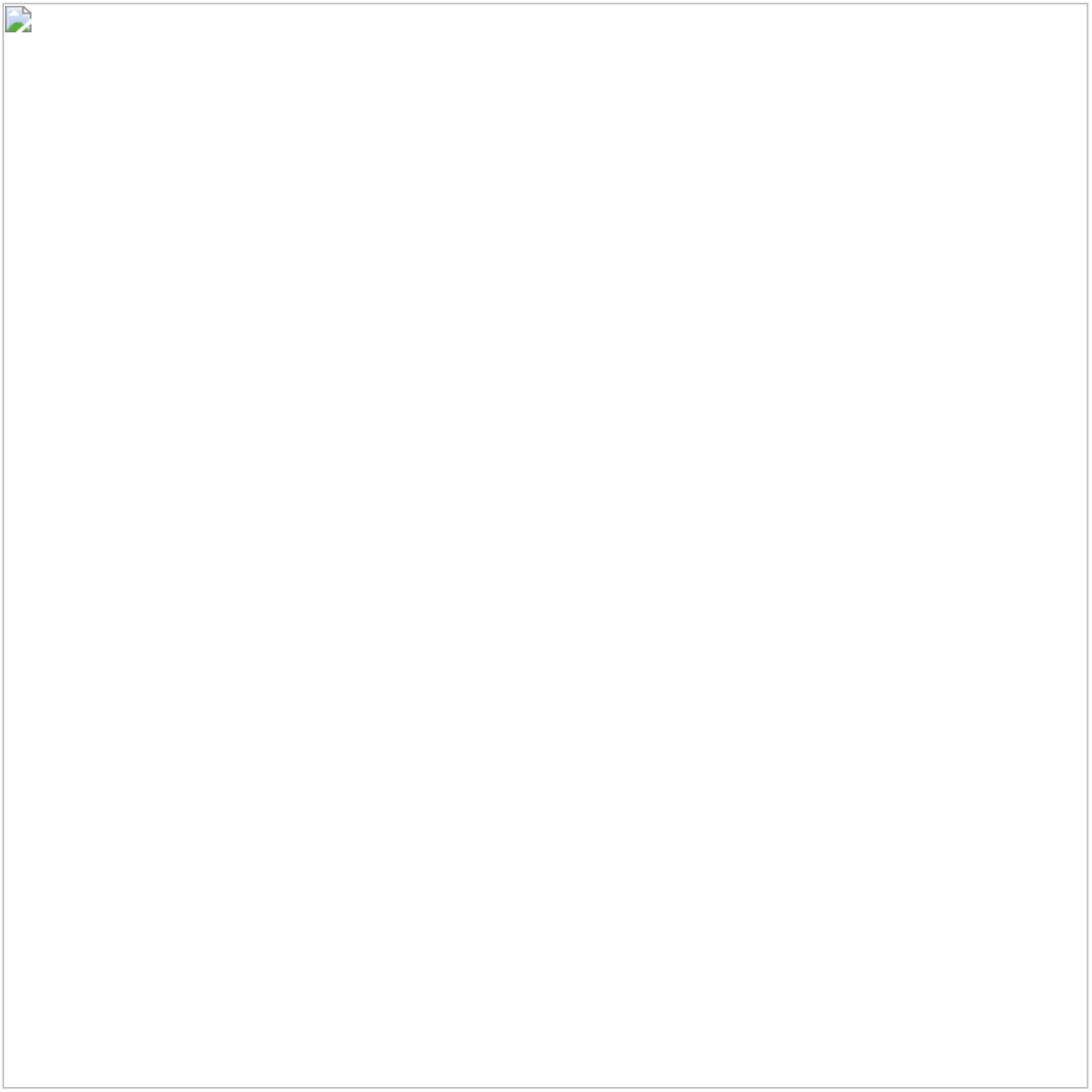
由于我们之前计算出了最后一层的delta误差

，通过上式，我们可以依次求得

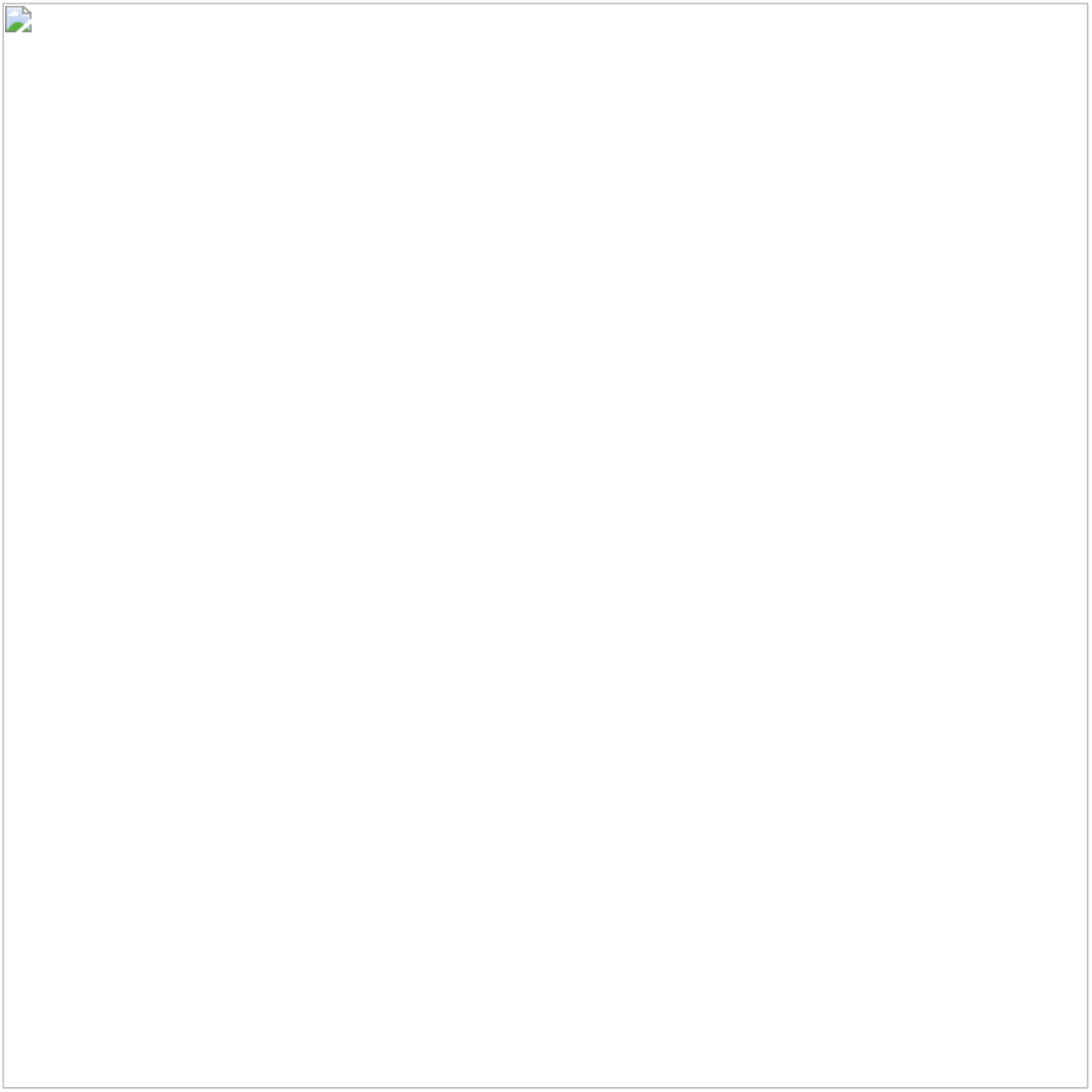
，

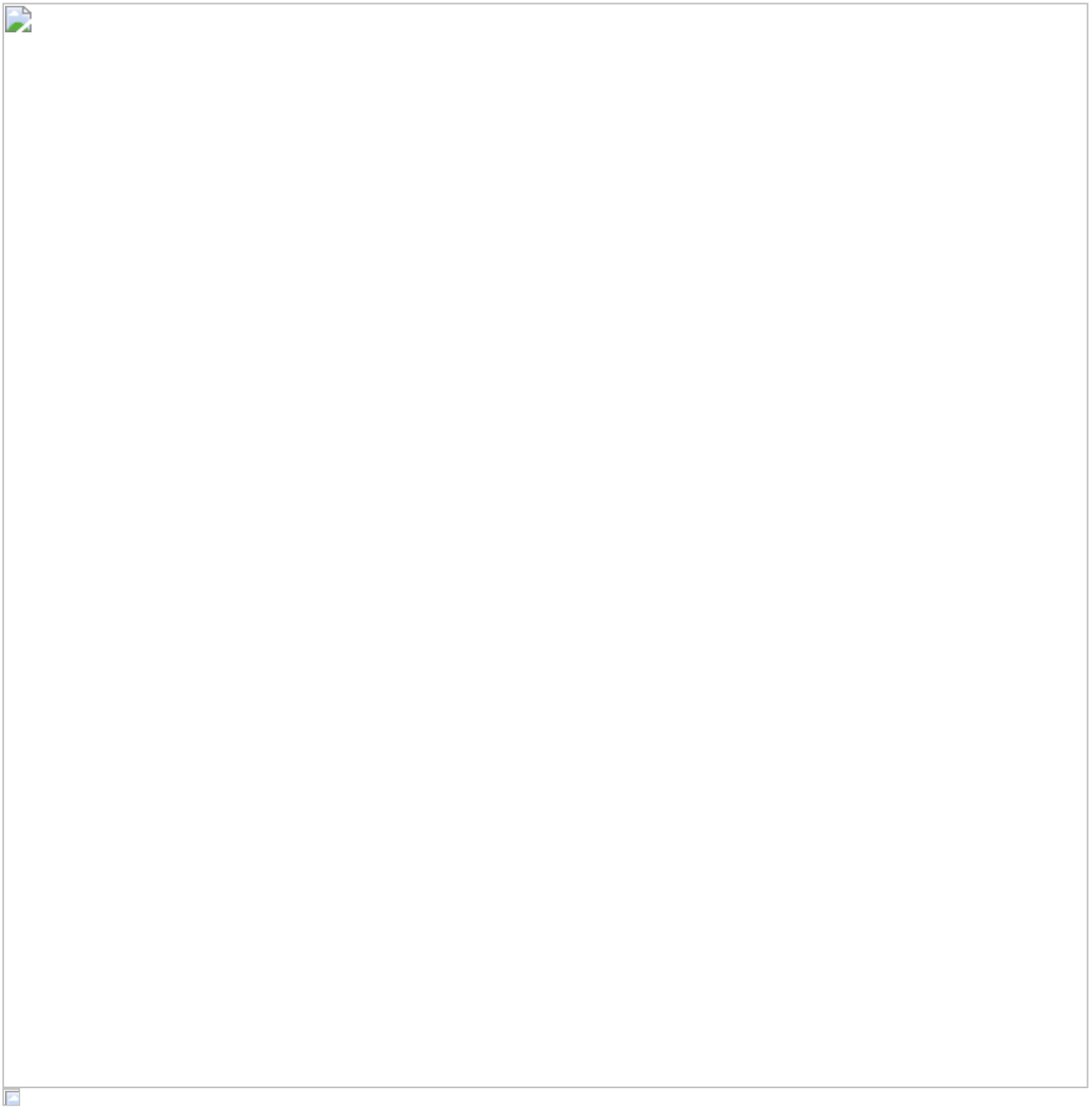
一直到第二层的delta误差

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。因此我们的计算到第二层截止。
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：




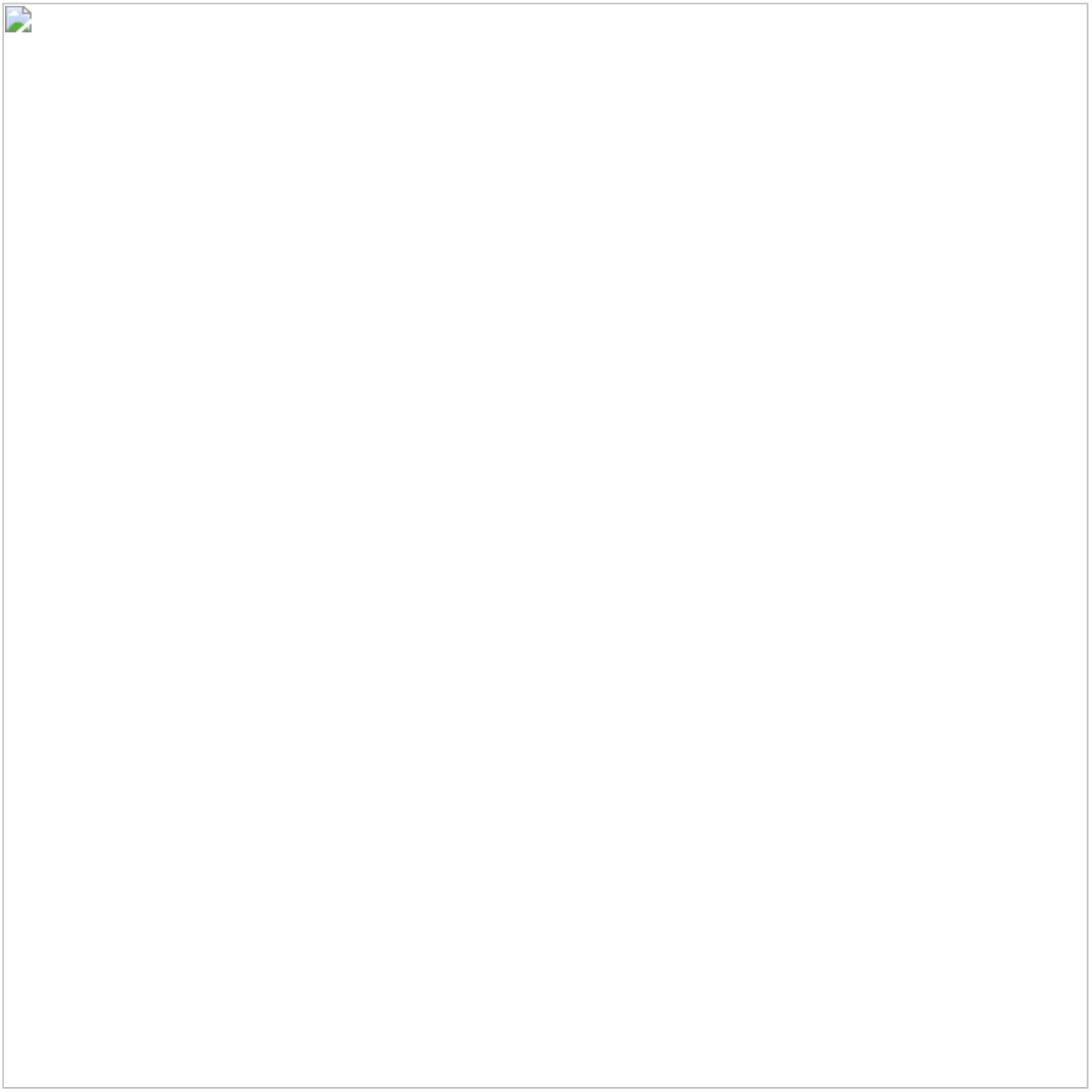
最后我们可以通过梯度下降法来对每一层的参数进行更新：

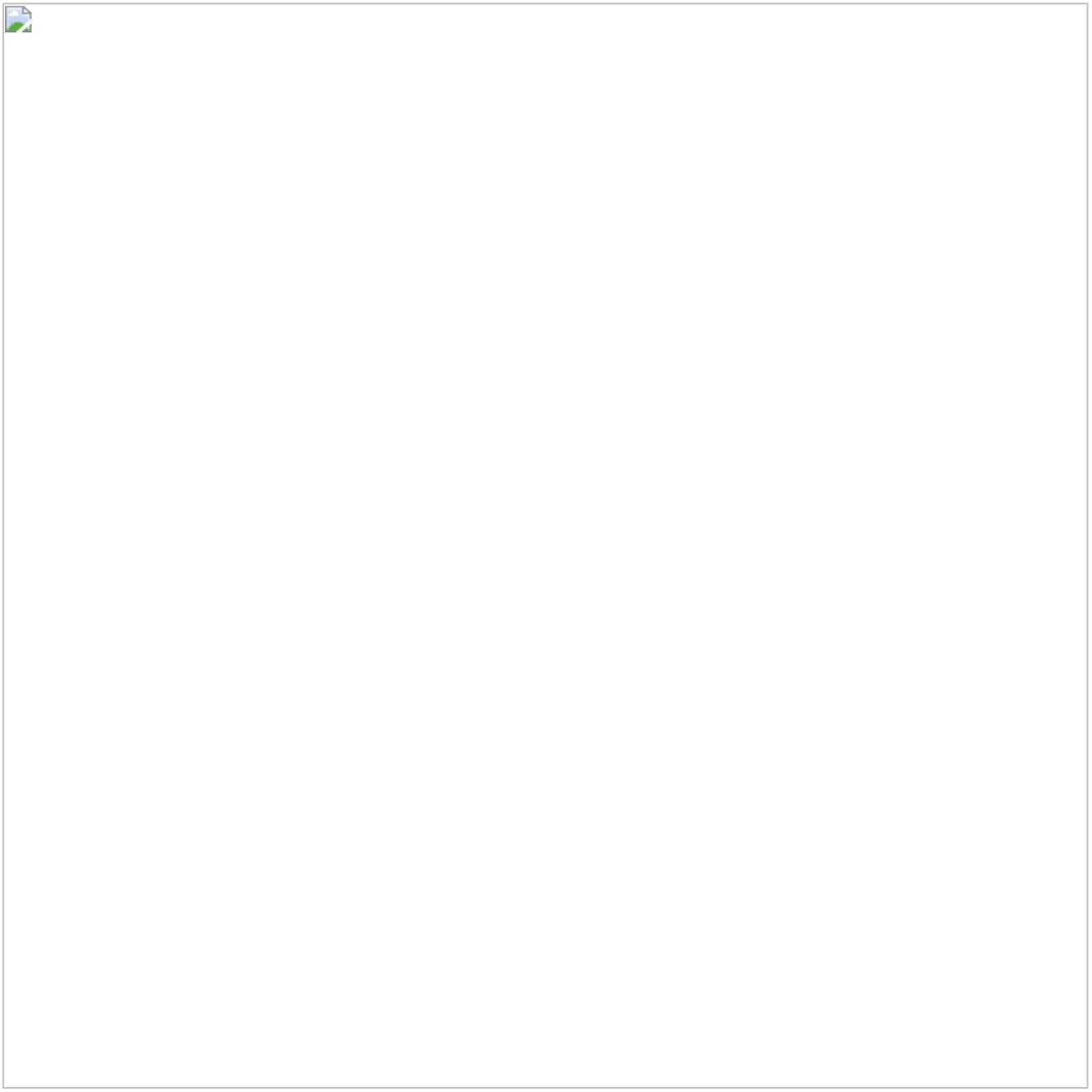




表示训练时的学习率。

在上述的分析中，我们只根据一组训练数据更新数据，而在一般的情况下，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每一个数据对应的误差梯度，最后再根据它们的平均值来对参数进行更新，即：





理论部分推导完了，让我们回顾一下，如何完成一个多层感知机的训练:

1. 对神经网络各层参数即各层的权重矩阵和偏置项进行初始化，

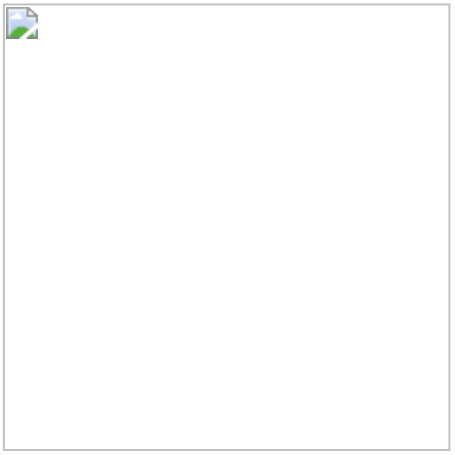
设置好训练的最大迭代次数，每个训练batch的大小，学习率

2. 从训练数据中取出一个batch的数据
3. 从该batch数据中取出一个数据，包括输入x以及对应的正确标注y
4. 将输入x送入神经网络的输入端，得到神经网络各层输出参数

和

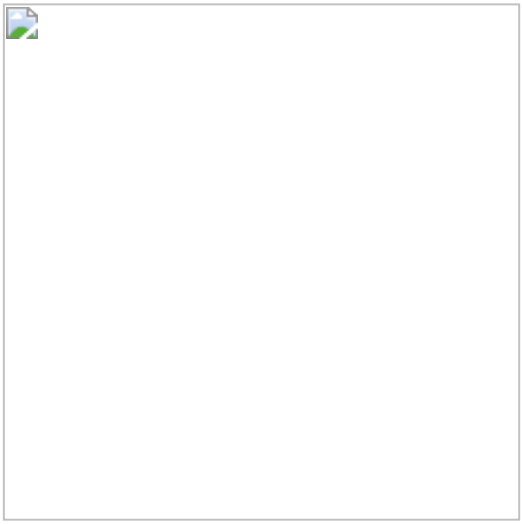
5. 根据神经网络的输出和标注值y计算神经网络的损失函数
6. 计算损失函数对输出层的delta误差

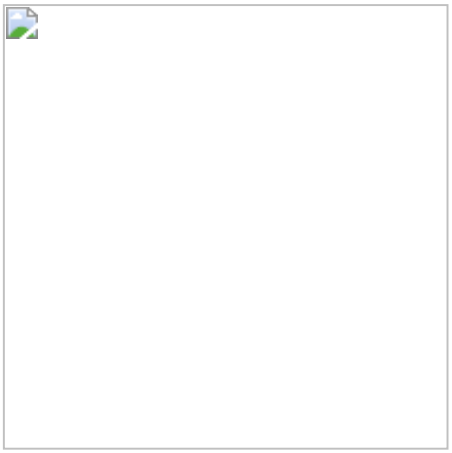
7. 利用相邻层之间delta误差的递推公式



求得每一层的delta误差

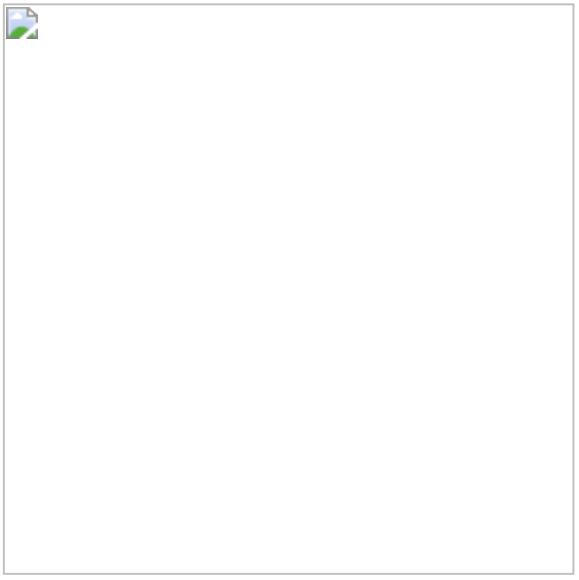
8. 利用每一层的delta误差求出损失函数对该层参数的导数

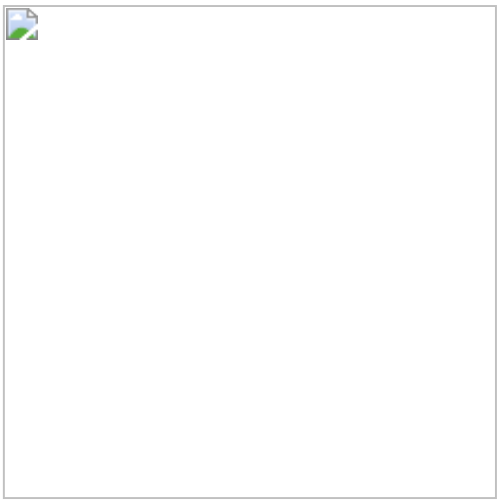




9. 将求得的导数加到该batch数据求得的导数之和上(初始化为0)，跳转到步骤3，直到该batch数据都训练完毕

10. 利用一个batch数据求得的导数之和，根据梯度下降法对参数进行更新





11. 跳转到步骤2，直到达到指定的迭代次数

参考：

[1]刘建平Pinard:深度神经网络（DNN）反向传播算法(BP)
cnblogs.com/pinard/p/64
[2] Neural Networks and Deep Learning by By Michael Nielsen
neuralnetworksanddeeplearning.com

▲长按加群

这么硬的干货分享，麻烦给我一个在在看

▲长按关注我们

麻烦给我一个在看！