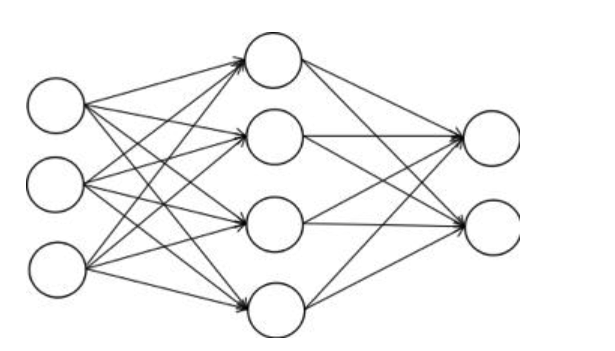
反向传播究极推导

我们来看一个简单的神经网络的例子，如下图所示：



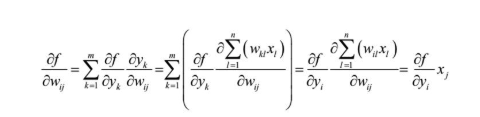
这个网络有三层。

* 第一层是输入层，对应的输入向量为x，有3个神经元，写成分量形式为(x1, x2,x3)，他不对数据做任何处理，直接原样送入下一层。
* 中间层有4个神经元，接受的输入数据为向量x，输出向量为y，写成分量形式为(y1, y2, y3,y4)。
* 第三层为输出层，接受的输入数据为向量y，输出向量为z，写成分量形式就是(z1, z2)。
* 第一层到第二层的权重矩阵为W(1)，第二层到第三层的权重矩阵为W(2)。权重矩阵的每一行都是一个权重向量，是上一层所有神经元到本层神经元的连接权重，这里的上标表示层数。
  + 比如W(1)为4\*3的矩阵，共四行，对应着第二层有4个神经元，计算y1时是：，所以说权重矩阵的每一行是一个权重矩阵。

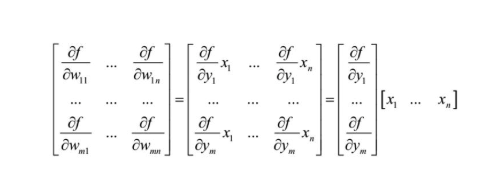
在推导之前，有几个重要的结论需要我们掌握一下：

* 结论1：

如下映射函数：y = wx, 其中y是m为向量，w是m\*n的矩阵，x是n维向量。假设有函数f(y)【可以把f()看做是激活函数】 ，那么我们如何根据函数y的梯度值计算函数对w的梯度值呢，根据链式法则，由于只和有关，和其他的( k不等于i) 无关，因此有：



也是就f对y中的每一个元素求偏导，y再对wij求偏导，但是只有当x的下标为j时，偏导才不为0，最终结果也就是,所以对于w中的所有元素，最后的偏导为：

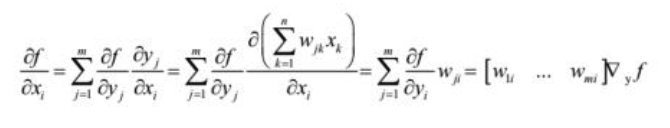


最后写成矩阵形式为：

总结，有两个嵌套公式，y=wx, f(y)，那么应用链式法则时，可以写成上述的矩阵公式。

* 结论2：

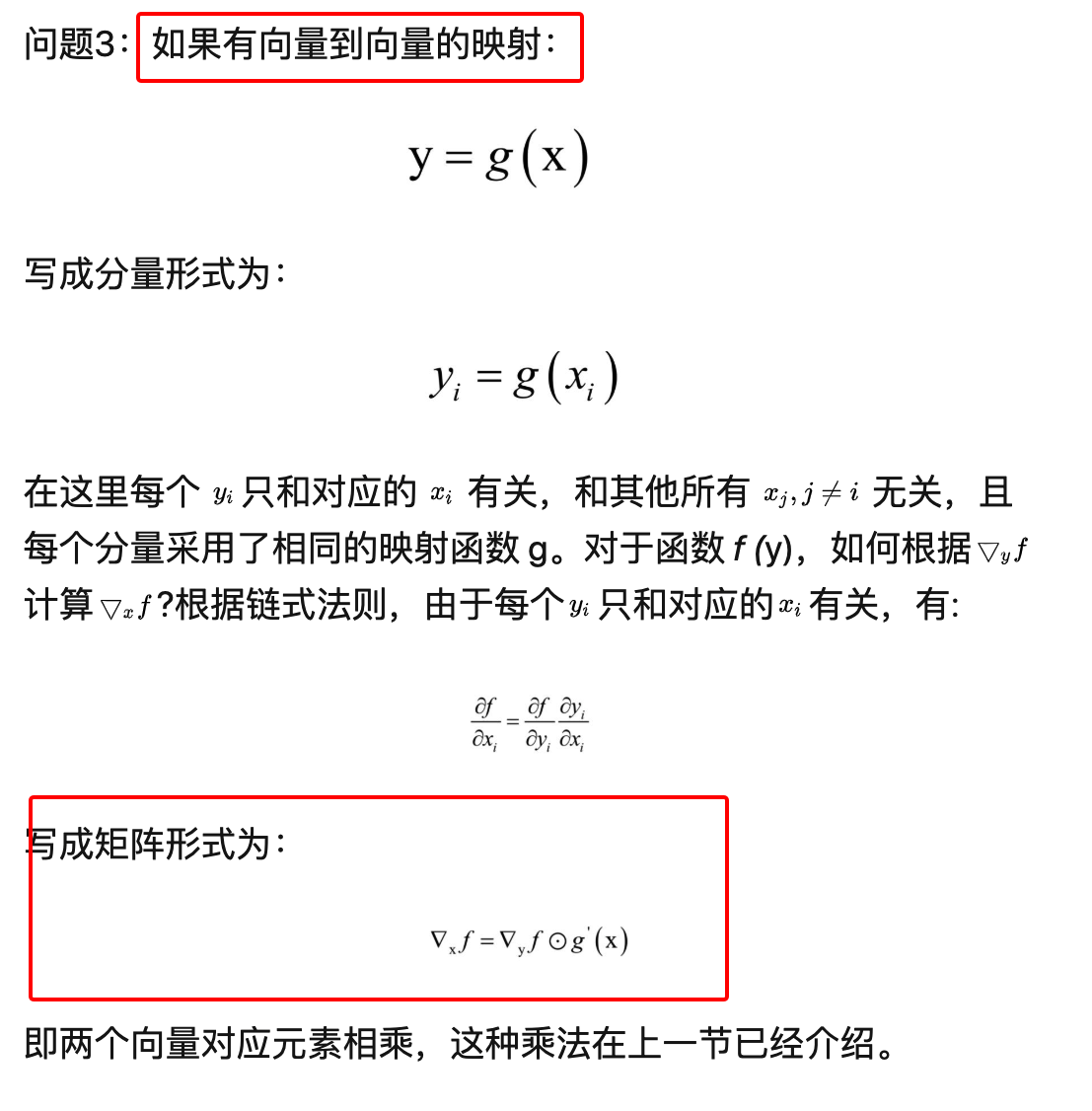
如果上式y=wx中，把w看成是常数，那么就可以把y看成是x的函数【结论1中把y看做是w的函数】，由于任意的xi和所有的yj都有关系【这里可以映射到上一层的神经元和下一层的神经元都有关】，根据链式法则则有：



写成矩阵形式则是：

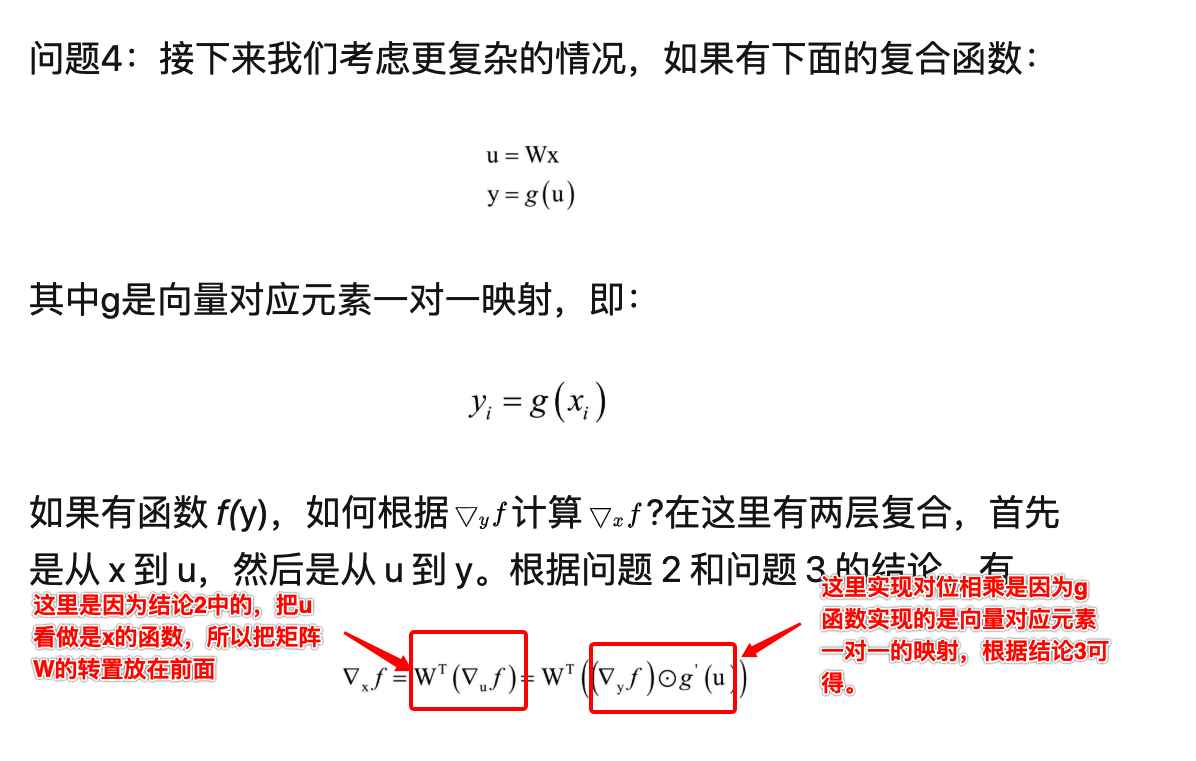
这是对称的，在计算函数对称时用矩阵w乘以向量x得到y，在求梯度时用矩阵w的转置乘以y的梯度得到x的梯度。

* 结论3

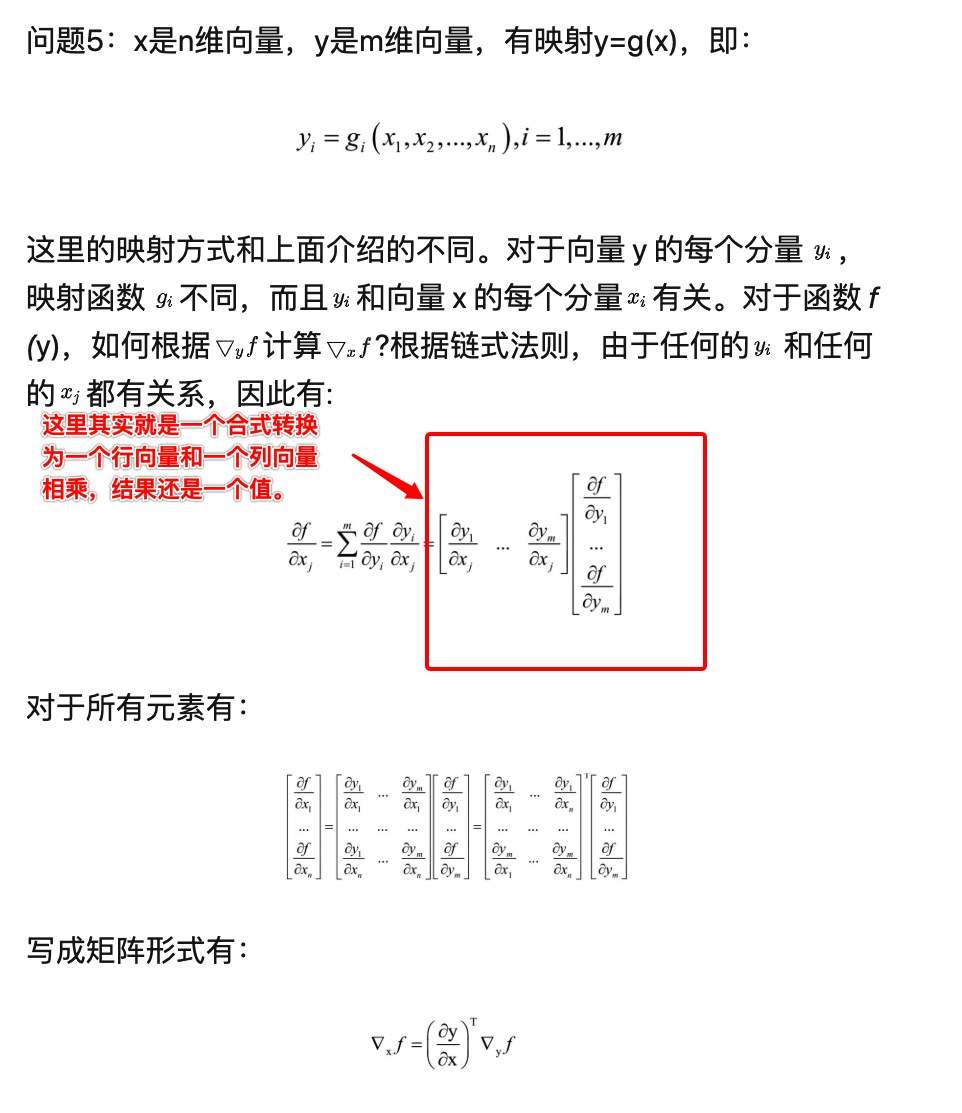


这里最后的形式其实是两个向量的对应元素相乘，也是因为yi只和xi有关。

* 结论4：



* 结论5



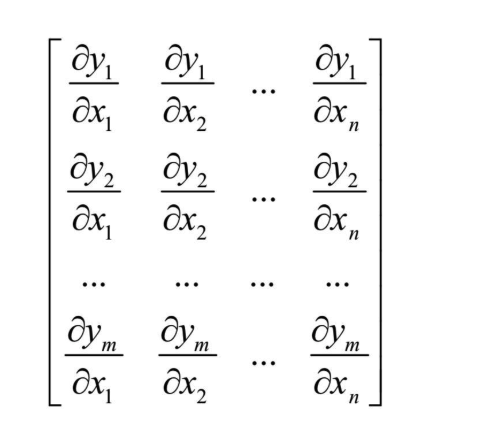
其中为雅克比矩阵。

雅克比矩阵的定义为：对于如下向量到向量的映射函数：

y = f (x)

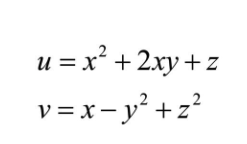
其中向量，向量，这个映射 写成分量形式为：

即输出向量的每个分量是输入向量的函数。雅克比矩阵的定义为：输出向量的每个分量对输入向量的每个分量的偏导数构成的矩阵， 相应的Hessian矩阵是输出向量的每个分量对输入向量的每个分量的2阶偏导构成的。

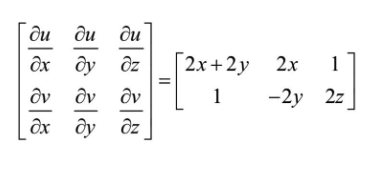


上图是一个m行n列的矩阵，每一行为一个多元函数的梯度。

对于如下函数：



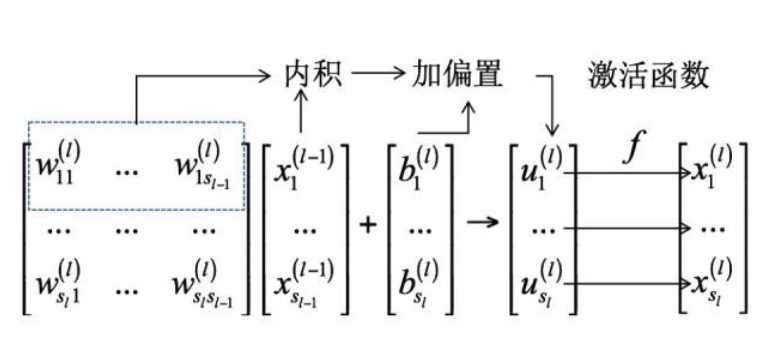
它的雅克比矩阵为：



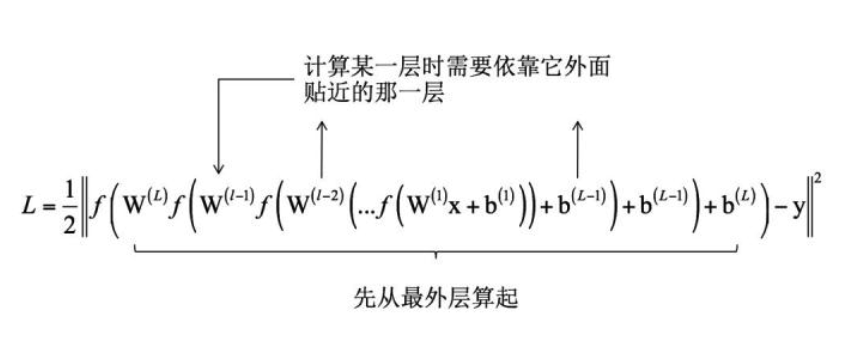
**下面介绍完整的反向传播算法**

根据上面的结论可以很方便的推导出神经网络的求导公式。假设神经网络有层，第l层的神经元的个数为。第l层从第l-1层接受的输入向量为，本层的权重矩阵为，偏置向量为，输出向量为，该层的输出可以写成如下矩阵形式：

其中是的矩阵【每一行都是一个权重，可以查看上文对权重矩阵的介绍】，和是维的向量。神经网络一个层的实现的变换如下图所示：

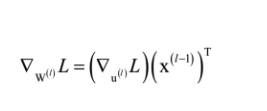


如果将神经网咯按照各层展开，最后得到一个深层的复合函数，将其代入欧氏距离损失函数，依然是一个关于各个层的权重矩阵和偏置向量的复合函数。

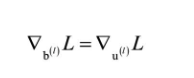


要计算某一层的权重矩阵和偏置向量的梯度，只能依赖于它紧贴着的外面那一层变量的梯度值，通过一次符合函数求导得到。

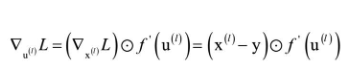
根据定义，和是目标函数的自变量，和可以看成是他们的函数。根据前面的结论，损失函数对权重矩阵的梯度为：



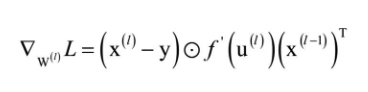
对偏置向量的梯度为：



现在的问题是，梯度是怎么计算得到的？我们分两种情况讨论，如果第l层是输出层，在这里只考虑单个样本的损失函数，根据前面推导的结论，这个梯度为：

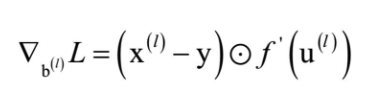


这就是输出层的神经元输出值与期望值之间的误差【注意这里求得是,是求损失函数对的梯度，不是对的梯度】。这样我们得到输出层权重的梯度为：

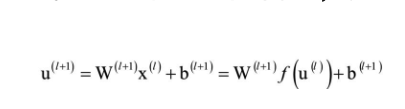


等号右边第一个乘法是向量对应元素乘【叫做点积或者叫做对位元素相乘】；第二个乘法是矩阵乘，在这里是列向量与行向量的乘积，结果是一个矩阵，尺寸刚好和权重矩阵相同。

损失函数对偏置项的梯度为：



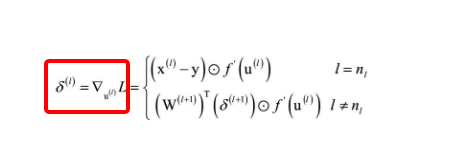
下面考虑第二种情况。如果第l层是隐含层，则有：



假设已经求出来了，根据前面的结论，有：

这是一个递推的关系，通过可以计算出，递推的终点是输出层，而输出层的梯度值我们之前已经计算出了。由于根据可以计算出和，因此可以计算出任意层权重和偏置的梯度值。

总结如下：



记住，我们反向求导过程中最重要的就是，而不是。

综上所述，这个反向传播的总体流程如下：

* 首先计算输出层的误差项，根据他得到的权重和偏置项的梯度，这是起点。
* 根据上面的递推公式，逐层向前，利用后一层的误差项计算出本层的误差项，从而得到本层权重和偏置项的梯度。

单个样本的反向传播算法在每次迭代时的流程为：

1. 正向传播，利用当前权重和偏置项，计算每一层对输入样本的输出值。
2. 反向传播，对输出层的每一个节点计算其误差：
3. 对于l = l-1,…2,1的各层，计算第l层的每个节点的误差：
4. 根据误差计算损失函数对权重的梯度值：

对偏置的梯度：

1. 利用梯度下降法更新权重和偏置：

实现时需要在正向传播时记住每一层的输入向量,本层的激活函数值。