

## 反向传播

神经网络之所以发展迅速，就是因为它的优化速度极快。这源于它的神经元拓补结构。下面我们就讲解一下：

## 基本原理

我们回顾神经网络的计算（为了方便表达，我们把上文中的 $W$ 直接记为 $W$ ）：

$$\begin{aligned}Z_1 &= X \cdot W_1 \\ A_1 &= f(Z_1) \\ Z_2 &= A_1 \cdot W_2 \\ A_2 &= f(Z_2) \\ Z_3 &= A_2 \cdot W_3 \\ A_3 &= f(Z_3)\end{aligned}$$

当然，网络的层数可以是任意的，但是我们就以3层为例。也就是说， $A_3$ 就是输出层的输出，我们也可以叫它 $\hat{Y}$ ，表示对于 $Y$ 的估计值。同时，我们也可以叫 $X$ 为 $A_0$ ，以达到形式上的统一。

最后假设我们定义loss function为

$$\mathcal{L}(\hat{Y}, Y) = \frac{1}{2} \| \hat{Y} - Y \|_2^2$$

在训练神经网络的过程中，我们的目标就是通过改变 $W$ 的值来让 $\hat{Y}$ 和 $Y$ 尽可能接近，也就是让损失函数 $\mathcal{L}$ 尽可能的小。

要通过 $W$ 减小 $\mathcal{L}$ ，我们就用到梯度下降。也就是说，我们需要计算 $\frac{d\mathcal{L}}{dW}$ 。

得益于神经网络的拓补结构（层状结构），我们可以轻而易举的利用链式法则计算梯度，比如

$$\frac{d\mathcal{L}}{dW_3} = \frac{d\mathcal{L}}{dA_3} \frac{dA_3}{dZ_3} \frac{dZ_3}{dW_3}$$

或者比如说

$$\frac{d\mathcal{L}}{dW_1} = \frac{d\mathcal{L}}{dA_3} \frac{dA_3}{dZ_2} \frac{dZ_2}{dA_2} \frac{dA_2}{dZ_1} \frac{dZ_1}{dA_1} \frac{dA_1}{dW_1}$$

由于每一部分都很简单，我们可以轻而易举的导出梯度。（注意，为了方便读者理解，这里的数学表达并不严谨。严谨的推导在最后）

## 简单推导

这里的推导只是展示反向传播的基本意思，数学表达上并不严谨。具体的推导在最后，可以选择忽略。

从上面的公式我们可以看出，这个梯度主要分成3部分：

- 从损失函数传递到输出层： $\frac{d\mathcal{L}}{dA_L}$
- 层内传播： $\frac{dA_i}{dA_{i-1}} = \frac{dA_i}{dZ_i} \frac{dZ_i}{dA_{i-1}}$
- 从层到权重： $\frac{dZ_i}{dW_i}$

当网络定义好了之后，每一部分都很简单。下面我们就推导它们。

### 损失函数到输出层

当我们定义

$$\mathcal{L}(A_L, Y) = \frac{1}{2} \| A_L - Y \|_2^2$$

之后，我们就可以得到：

$$\frac{d\mathcal{L}}{dA_L} = A_L - Y$$

这里的 $Y$ 显然是已知的，而 $A_L$ 的值其实在计算过程中也是已经算出来了的。

当然，只要定义的损失函数可导，我们都可以计算梯度，例如我们现在用cross entropy当作损失函数来对来对分类任务进行训练：

$$\mathcal{L} = -Y \log A_L - (1 - Y) \log(1 - A_L)$$

先不考虑它的具体意义，只考虑数学推导，那么我们可以得到

$$\frac{d\mathcal{L}}{dA_L} = -\frac{Y}{A_L} + \frac{1 - Y}{1 - A_L}$$

这里的除法指的是每个元素的计算，而不是“矩阵除法”。

### 层与层之间的传播

$$\frac{dA_i}{dA_{i-1}} = \frac{dA_i}{dZ_i} \frac{dZ_i}{dA_{i-1}}$$

对于第一部分，当我们定义了激活函数之后，就可以轻易得到，我们以sigmoid为例：

$$A_i = \frac{1}{1 + e^{-Z_i}}$$

那么

$$\begin{aligned}\frac{dA_i}{dZ_i} &= -\left(\frac{1}{e^{-Z_i} + 1}\right)^2 e^{-Z_i}(-1) \\ &= \frac{e^{-Z_i}}{(e^{-Z_i} + 1)^2} \\ &= \frac{e^{-Z_i} + 1 - 1}{(e^{-Z_i} + 1)^2} \\ &= \frac{e^{-Z_i} + 1}{(e^{-Z_i} + 1)^2} - \frac{1}{(e^{-Z_i} + 1)^2} \\ &= \frac{1}{e^{-Z_i} + 1} - \frac{1}{(e^{-Z_i} + 1)^2} \\ &= A_i - A_i^2 \\ &= A_i(1 - A_i)\end{aligned}$$

注意这里的乘法也是元素相乘，而不是矩阵计算。我们一般把它记为

$$\frac{dA_i}{dZ_i} = A_i \circ (1 - A_i)$$

对于第二部分，因为

$$Z_i = A_{i-1} \cdot W_i$$

我们可以轻松的得到

$$\frac{dZ_i}{dA_{i-1}} = W_i$$

那么从第 $i$ 层的输出 $A_i$ 到第 $i - 1$ 层的输出 $A_{i-1}$ 就是

$$\frac{dA_i}{dA_{i-1}} = \frac{dA_i}{dZ_i} \frac{dZ_i}{dA_{i-1}} = A_i \circ (1 - A_i) \circ W_i$$

这里的 $A_{i-1}$ 也会在正向传播的时候算出来，而 $W_i$ 则是模型当前的参数值。这两个都是已知的。

注意这里都是元素相乘，详细推导在后面，有兴趣可以看。

### 从层到权重的传递

由于

$$Z_i = A_{i-1} \cdot W_i$$

所以

$$\frac{dZ_i}{dW_i} = A_{i-1}$$

这里的 $A_{i-1}$ 也会在正向传播的时候算出来，也是已知量。

## 总结

这一章节我们推导了梯度从损失函数传递到权重的过程。可以看出，当模型一旦确定，这些梯度的推导公式就确定了。唯一要做的就是正向传播一次，计算出每一层的 $Z_i$ 和 $A_i$ ，然后再借助目标数据 $Y$ 和模型当前的参数值 $W_i$ ，就可以通过简单的矩阵乘法得到 $W_i$ 的梯度。这是一种非常高效的方法，因此也使得优化上万千参数成为可能。

## 详细推导

这一部分有兴趣的读者可以看。

下面的推导均建立在sigmoid作为激活函数，cross entropy作为损失函数的网络。但是正如上面所讲的，无论怎么更改函数，只要把更改部分的梯度表达式推导出来，就可以进行局部替换。但是整体的反向传播过程不变。

关于偏差 $b$ ，它往和 $W$ 合在一起，然后把 $X$ 扩展一列出来。在下面的推导中，我们把它分开写。如果想要改成 $b$ 和 $W$ 合起来的形式，只需要忽略与 $b$ 有关的内容即可。

下面我们从1个神经元，1个输入数据，2个分类的最简单情况，逐步扩展到 $I$ 个神经元， $E$ 个数据， $K$ 个分类的通用情况。

### 1神经元，1数据，2分类

这种情况下，所有的值都是单个的实数标量。正向传播：

$a_1 = x$	$\mathbb{R}$
$z_l = w_l a_{l-1} + b_l$	$\mathbb{R}$
$a_l = \sigma(z_l)$	$\mathbb{R}$
$z_L = w_L a_{L-1} + b_L$	$\mathbb{R}$
$a_L = \sigma(z_L)$	$\mathbb{R}$
$\mathcal{L} = -y \log a_L - (1 - y) \log(1 - a_L)$	$\mathbb{R}$

反向传播

$\frac{\partial \mathcal{L}}{\partial a_L} = -\frac{y}{a_L} + \frac{1-y}{1-a_L}$	$\mathbb{R}$
$\frac{\partial a_L}{\partial z_L} = a_L(1 - a_L)$	$\mathbb{R}$
$\frac{\partial \mathcal{L}}{\partial z_L} = \frac{\partial \mathcal{L}}{\partial a_L} \frac{\partial a_L}{\partial z_L} = a_L - y$	$\mathbb{R}$
$\frac{\partial z_{l+1}}{\partial a_l} = \frac{\partial z_{l+1}}{\partial a_l} \frac{\partial a_l}{\partial z_l} = w_{l+1} \cdot a_l \cdot (1 - a_l)$	$\mathbb{R}$
$\frac{\partial \mathcal{L}}{\partial w_l} = \frac{\partial \mathcal{L}}{\partial z_{l+1}} \frac{\partial z_{l+1}}{\partial a_l}$	$\mathbb{R}$
$\frac{\partial z_l}{\partial a_l} = a_{l-1}$	$\mathbb{R}$
$\frac{\partial a_l}{\partial w_l} = 1$	$\mathbb{R}$
$\frac{\partial \mathcal{L}}{\partial w_{l+1}} = \frac{\partial \mathcal{L}}{\partial z_{l+1}} \frac{\partial z_{l+1}}{\partial w_{l+1}} = \frac{\partial \mathcal{L}}{\partial z_{l+1}} a_l$	$\mathbb{R}$
$\frac{\partial \mathcal{L}}{\partial w_{l+1}} = \frac{\partial \mathcal{L}}{\partial z_{l+1}} \frac{\partial z_{l+1}}{\partial w_{l+1}} = \frac{\partial \mathcal{L}}{\partial z_{l+1}}$	$\mathbb{R}$

用上述的公式就可以求出任意 $w_l$ 和 $b_l$ 的梯度了。

### I神经元，1个数据，2个分类

这里需要一个示意图来帮助推导：

层与层之间的传递：

$$\frac{\partial z_{l+1}}{\partial z_l} = \left[ \frac{\frac{\partial z_{l+1}}{\partial z_l^1}}{\frac{\partial z_{l+1}}{\partial z_l^2}} \frac{\frac{\partial z_{l+1}}{\partial z_l^2}}{\frac{\partial z_{l+1}}{\partial z_l^3}} \frac{\frac{\partial z_{l+1}}{\partial z_l^3}}{\frac{\partial z_{l+1}}{\partial z_l^4}} \right] = \left[ \frac{\frac{\partial z_{l+1}^1}{\partial z_l^1} \frac{\partial z_{l+1}^1}{\partial z_l^2} \frac{\partial z_{l+1}^1}{\partial z_l^3}}{\frac{\partial z_{l+1}^2}{\partial z_l^1} \frac{\partial z_{l+1}^2}{\partial z_l^2} \frac{\partial z_{l+1}^2}{\partial z_l^3}} \frac{\frac{\partial z_{l+1}^3}{\partial z_l^1} \frac{\partial z_{l+1}^3}{\partial z_l^2} \frac{\partial z_{l+1}^3}{\partial z_l^3}}{\frac{\partial z_{l+1}^4}{\partial z_l^1} \frac{\partial z_{l+1}^4}{\partial z_l^2} \frac{\partial z_{l+1}^4}{\partial z_l^3}} \right] \in \mathbb{R}^{n_{l+1} \times n_l} \quad (1)$$

$$= \left[ \frac{W_{l+1}^{1,1} \frac{\partial z_{l+1}^1}{\partial z_l^1}}{W_{l+1}^{2,1} \frac{\partial z_{l+1}^2}{\partial z_l^1}} \frac{W_{l+1}^{1,2} \frac{\partial z_{l+1}^1}{\partial z_l^2}}{W_{l+1}^{2,2} \frac{\partial z_{l+1}^2}{\partial z_l^2}} \frac{W_{l+1}^{1,3} \frac{\partial z_{l+1}^1}{\partial z_l^3}}{W_{l+1}^{2,3} \frac{\partial z_{l+1}^2}{\partial z_l^3}} \right] = \left[ W_{l+1}^{1,1} \frac{W_{l+1}^{1,2}}{W_{l+1}^{2,1}} \frac{W_{l+1}^{1,3}}{W_{l+1}^{2,2}} \right] \circ \left[ \frac{\partial a_l^1}{\partial z_l^1} \frac{\partial a_l^1}{\partial z_l^2} \frac{\partial a_l^1}{\partial z_l^3} \right] \quad (2)$$

$$= W_{l+1}^{1,1} \circ \left[ a_l^1(1 - a_l^1) \right] \frac{a_l^2(1 - a_l^2)}{a_l^3(1 - a_l^3)} \frac{a_l^3(1 - a_l^3)}{a_l^4(1 - a_l^4)} \quad (3)$$

$$= W_{l+1}^1 \circ a_l \circ (1 - a_l) \quad (4)$$

这里又牵扯到2个额外的知识。第一个就是“圈乘”( $\circ$ )，学名叫舒尔积或者阿达玛积，指的是两个维度一样的矩阵的每个元素对应相乘。

第二个点就是broad casting，也就是说，当两个变量运算时，如果维度缺失，一些编程语言会自动补充缺失维度。再缺失的维度上会自动复制，比如：

$$\begin{bmatrix} W_{l+1}^{1,1} & W_{l+1}^{1,2} & W_{l+1}^{1,3} \\ W_{l+1}^{2,1} & W_{l+1}^{2,2} & W_{l+1}^{2,3} \end{bmatrix} \circ \begin{bmatrix} \frac{\partial a_l^1}{\partial z_l^1} & \frac{\partial a_l^2}{\partial z_l^2} & \frac{\partial a_l^3}{\partial z_l^3} \end{bmatrix}$$

是一个 $3 \times 2$ 矩阵和 $3 \times 1$ 矩阵的舒尔积，那么第二个矩阵会被复制乘 $3 \times 2$ ，也就相当于：

$$\begin{bmatrix} W_{l+1}^{1,1} & W_{l+1}^{1,2} & W_{l+1}^{1,3} \\ W_{l+1}^{2,1} & W_{l+1}^{2,2} & W_{l+1}^{2,3} \end{bmatrix} \circ \begin{bmatrix} \frac{\partial a_l^1}{\partial z_l^1} & \frac{\partial a_l^1}{\partial z_l^2} & \frac{\partial a_l^1}{\partial z_l^3} \\ \frac{\partial a_l^2}{\partial z_l^1} & \frac{\partial a_l^2}{\partial z_l^2} & \frac{\partial a_l^2}{\partial z_l^3} \\ \frac{\partial a_l^3}{\partial z_l^1} & \frac{\partial a_l^3}{\partial z_l^2} & \frac{\partial a_l^3}{\partial z_l^3} \end{bmatrix}$$

这里有趣的一点在于出现了舒尔积的运算。相比于矩阵乘积，舒尔积的运算其实是稀疏的，因为只有对应元素相乘，也就是说一乘数矩阵的一个元素只会影响结果中的一个元素。之所以会这样，是因为神经网络的拓补结构，例如图中的 $W_{l+1}^{1,1}$ ， $W_{l+1}^{1,2}$ ， $W_{l+1}^{1,3}$ 元素只和 $z_{l+1}^1$ 相关，而和 $z_{l+1}^2$ 无关。这样，当矩阵

$$Z_{l+1} = \begin{bmatrix} z_{l+1}^1 \\ z_{l+1}^2 \end{bmatrix}$$

当 $Z_{l+1}$ 对 $W_{l+1}^{1,1}$ ， $W_{l+1}^{1,2}$ ， $W_{l+1}^{1,3}$ 求导时，就等价于只有 $z_{l+1}^1$ 对它们求导，其他的都是0。

当 $Z_{l+1}$ 对 $W_{l+1}^{2,1}$ ， $W_{l+1}^{2,2}$ ， $W_{l+1}^{2,3}$ 求导时，就等价于只有 $z_{l+1}^2$ 对它们求导，其他的都是0。

同理， $A_l$ 对 $Z_l$ 求导时，相当于 $a_l^1$ 对 $z_l^1$ 单独求导， $a_l^2$ 对 $z_l^2$ 单独求导， $a_l^3$ 对 $z_l^3$ 单独求导，而不是 $A_l$ 的每一个元素对 $Z_l$ 中的每一个元素分别单独求导。（本质上应该是这样的，但是由于网络的结构，非对应元素的求导都为0）。

然后我们推导损失函数到输出层的公式

$$\frac{\partial \mathcal{L}}{\partial z_l} = \left[ \frac{\partial \mathcal{L}}{\partial z_l^1} \frac{\partial \mathcal{L}}{\partial z_l^2} \frac{\partial \mathcal{L}}{\partial z_l^3} \right] = \left[ \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial z_{l+1}^1}{\partial z_l^1} + \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial z_{l+1}^2}{\partial z_l^1} \frac{\partial \mathcal{L}}{\partial z_{l+1}^3} \frac{\partial z_{l+1}^3}{\partial z_l^1} + \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial z_{l+1}^1}{\partial z_l^2} + \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial z_{l+1}^2}{\partial z_l^2} + \frac{\partial \mathcal{L}}{\partial z_{l+1}^3} \frac{\partial z_{l+1}^3}{\partial z_l^2} \right] \quad (5)$$

$$= \left[ \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial \mathcal{L}}{\partial z_{l+1}^3} \right] \cdot \begin{bmatrix} \frac{\partial z_{l+1}^1}{\partial z_l^1} \frac{\partial z_{l+1}^1}{\partial z_l^2} \frac{\partial z_{l+1}^1}{\partial z_l^3} \\ \frac{\partial z_{l+1}^2}{\partial z_l^1} \frac{\partial z_{l+1}^2}{\partial z_l^2} \frac{\partial z_{l+1}^2}{\partial z_l^3} \\ \frac{\partial z_{l+1}^3}{\partial z_l^1} \frac{\partial z_{l+1}^3}{\partial z_l^2} \frac{\partial z_{l+1}^3}{\partial z_l^3} \end{bmatrix} \in \mathbb{R}^{1 \times n_l} \quad (6)$$

$$= \frac{\partial \mathcal{L}}{\partial z_{l+1}} \cdot \frac{\partial z_{l+1}}{\partial z_l} \quad (7)$$

我们再推导损失函数到参数的传播：

$$\frac{\partial \mathcal{L}}{\partial W_{l+1}} = \left[ \frac{\partial \mathcal{L}}{\partial W_{l+1}^{1,1}} \frac{\partial \mathcal{L}}{\partial W_{l+1}^{1,2}} \frac{\partial \mathcal{L}}{\partial W_{l+1}^{1,3}} \right] = \left[ \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial z_{l+1}^1}{\partial W_{l+1}^{1,1}} \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial z_{l+1}^2}{\partial W_{l+1}^{1,2}} \frac{\partial \mathcal{L}}{\partial z_{l+1}^3} \frac{\partial z_{l+1}^3}{\partial W_{l+1}^{1,3}} \right] \in \mathbb{R}^{n_l \times n_{l+1}} \quad (8)$$

$$= \left[ \frac{\frac{\partial \mathcal{L}}{\partial z_{l+1}^1} a_l^1 \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial \mathcal{L}}{\partial z_{l+1}^3}}{a_l^2 \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} a_l^3 \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial \mathcal{L}}{\partial z_{l+1}^3}} \right] = \begin{bmatrix} a_l^1 \\ a_l^2 \\ a_l^3 \end{bmatrix} \left[ \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \right] \quad (9)$$

$$= a_l^\top \frac{\partial \mathcal{L}}{\partial z_{l+1}} \quad (10)$$

$$\frac{\partial \mathcal{L}}{\partial b_{l+1}} = \left[ \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \right] = \left[ \frac{\partial \mathcal{L}}{\partial z_{l+1}^1} \frac{\partial z_{l+1}^1}{\partial b_{l+1}} \frac{\partial \mathcal{L}}{\partial z_{l+1}^2} \frac{\partial z_{l+1}^2}{\partial b_{l+1}} \right] = \frac{\partial \mathcal{L}}{\partial z_{l+1}} \in \mathbb{R}^{1 \times n_{l+1}} \quad (11)$$

### I神经元，1个数据，K个分类

对于 $K$ 类的分类问题， $y$ 将是一个One-Hot encoding成的 $K$ 维变量，比如当 $K = 4$ 时：

$$\begin{aligned}y &= [1, 0, 0, 0] \\ y &= [0, 1, 0, 0] \\ y &= [0, 0, 1, 0] \\ y &= [0, 0, 0, 1]\end{aligned}$$

分别代表第1类，第2类，第3类和第4类。

交叉熵（cross entropy）损失函数就是：

$$\mathcal{L} = \sum_k -y_k \log a_{Lk} - (1 - y_k) \log(1 - a_{Lk})$$

其中下标 $k = 1, \dots, K$ 表示 $y$ 和 $a_L$ 的第几个元素。cross entropy意义是让 $y_k = 0$ 的项对应的输出尽可能减小，让 $y_k = 1$ 的项对应的输出尽可能增大。那么可以得到：

$$\begin{aligned}\mathcal{L} &= \sum_k -y_k \log a_{Lk} - (1 - y_k) \log(1 - a_{Lk}) \\ &= \sum_k -y_k \log a_{Lk} - \sum_k (1 - y_k) \log(1 - a_{Lk})\end{aligned}$$

$$= [y_1, y_2, y_3, y_4] \cdot \begin{bmatrix} \log a_{L1} \\ \log a_{L2} \\ \log a_{L3} \\ \log a_{L4} \end{bmatrix} - [1 - y_1, 1 - y_2, 1 - y_3, 1 - y_4] \cdot \begin{bmatrix} \log(1 - a_{L1}) \\ \log(1 - a_{L2}) \\ \log(1 - a_{L3}) \\ \log(1 - a_{L4}) \end{bmatrix}$$

$$= y \log a_L^\top - (1 - y) \log(1 - a_L)^\top$$

因此，损失函数传递到输出层的公式就是

$$\frac{\partial \mathcal{L}}{\partial z_L} = \left[ \frac{\partial \mathcal{L}}{\partial z_L^1} \frac{\partial \mathcal{L}}{\partial z_L^2} \frac{\partial \mathcal{L}}{\partial z_L^3} \frac{\partial \mathcal{L}}{\partial z_L^4} \right] = \left[ \frac{\partial \mathcal{L}}{\partial z_{L+1}^1} \frac{\partial z_{L+1}^1}{\partial z_L^1} \frac{\partial \mathcal{L}}{\partial z_{L+1}^2} \frac{\partial z_{L+1}^2}{\partial z_L^1} \frac{\partial \mathcal{L}}{\partial z_{L+1}^3} \frac{\partial z_{L+1}^3}{\partial z_L^1} \frac{\partial \mathcal{L}}{\partial z_{L+1}^4} \frac{\partial z_{L+1}^4}{\partial z_L^1} \right] \quad (12)$$

$$= \left[ \left( -\frac{y^1}{a_L^1} + \frac{1 - y^1}{1 - a_L^1} \right) a_L^1(1 - a_L^1) \right]^\top \quad (13)$$

$$= \left[ \left( -\frac{y^2}{a_L^2} + \frac{1 - y^2}{1 - a_L^2} \right) a_L^2(1 - a_L^2) \right]^\top \quad (14)$$

$$= \left[ \left( -\frac{y^3}{a_L^3} + \frac{1 - y^3}{1 - a_L^3} \right) a_L^3(1 - a_L^3) \right]^\top \quad (15)$$

$$= \left[ a_L^1 - y^1 \quad a_L^2 - y^2 \quad a_L^3 - y^3 \quad a_L^4 - y^4 \right]^\top \quad (16)$$

$$= a_L - y \in \mathbb{R}^{1 \times K} \quad (17)$$

由于这一小节相对于上一小节只是把输出从1个变成 $K$ 个，而隐藏层和输入层没有任何变化，因此， $\frac{dZ_{l+1}}{dZ_l}$ 以及 $\frac{dZ_l}{dW_l}$ 的公式没有任何变化。

### I神经元，E个数据，K个分类

这就是一般的神经网络了。为了方便直观的推导，我们假设有 $E = 2$ 个数据，并且仍然假设 $K = 4$ ，那么：

$$X = \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} \in \mathbb{R}^{E \times M}, \quad Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{M \times K}$$

同时，对于第 $l$ 层的加权求和和激活之后的变量是：

$$A_l = \begin{bmatrix} a_l^{(1)} \\ a_l^{(2)} \end{bmatrix} \in \mathbb{R}^{M \times n_l}, \quad Z_l = \begin{bmatrix} z_l^{(1)} \\ z_l^{(2)} \end{bmatrix} \in \mathbb{R}^{M \times n_l}$$

其中上标 $(e) = (1), (2), \dots, (E)$ 表示第几个数据。

那么损失函数就是：

$$\mathcal{L} = \frac{1}{E} \sum_e \mathcal{L}^{(e)}$$

那么，展开之后就是

$$\mathcal{L} = \frac{1}{E} \sum_e \mathcal{L}^{(e)}$$

$$= \frac{1}{E} \sum_e y^{(e)} \log a_L^{(e)\top} - (1 - y^{(e)}) \log(1 - a_L^{(e)})^\top$$

再对每一个类 $k = 1, \dots, K$ 展开：

$$\frac{1}{E} \sum_e \sum_k y_k^{(e)} \log a_{Lk}^{(e)} - (1 - y_k^{(e)}) \log(1 - a_{Lk}^{(e)})$$

$$= \frac{1}{E} \sum_{(e,k)} -Y \circ \log A_L - (1 - Y) \circ \log(1 - A_L)$$

这里就可以直接用 $A_L$ 和 $Y$ 来计算损失函数，而不需要逐个数据迭代的运算。要注意，变化前后在数学意义上一样，但是在代码中，矩阵的直接运算往往比循环要快得多。这里面牵扯到许多底层的优化和并行运算的问题。

注意这里从形式上比之前多了一个 $\frac{1}{E}$ 。

因此，从损失函数传递到输出层的公式就是：

$$\frac{\partial \mathcal{L}}{\partial Z_L} = \left[ \left( \frac{\partial \mathcal{L}}{\partial Z_L} \right)^{(1)} \right. \left. \left( \frac{\partial \mathcal{L}}{\partial Z_L} \right)^{(2)} \right] \in \mathbb{R}^{E \times K}$$