## 神经网络的梯度下降

假设神经网络的参数为

$$\theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots\}$$

而我们如果要将参数按照梯度下降的方法对每个参数进行迭代如下

$$\nabla L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial w_1} \\ \frac{\partial L(\theta)}{\partial w_2} \\ \cdots\cdot \\ \frac{\partial L(\theta)}{\partial b_1} \\ \frac{\partial L(\theta)}{\partial b_2} \\ \cdots \end{bmatrix}$$

当有成百上千个参数的时候，此时的计算量将会特别大，因此，为了有效的计算梯度下降，我们推荐使用**反向传播算法 (Back Propagation )**

## 链式法则 Chain Rule

其实就是复合函数的求导法则，只是在多元函数求导的时候需要多多注意。

**Case 1**

$$y = g(x) \qquad z = h(y)$$

$$\Delta x \to \Delta y \to \Delta z \qquad \frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

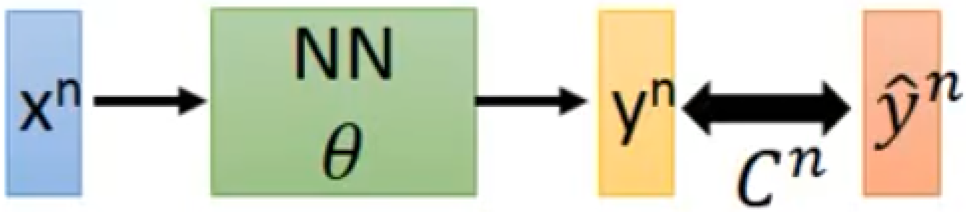
**Case 2**

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$

$$\begin{tikzcd} & \Delta x \arrow[dr] & \\ \Delta s \arrow[ur] \arrow[dr] & & \Delta z \\ & \Delta y \arrow[ur] & \end{tikzcd}$$

$$\frac{dz}{ds} = \frac{\partial z}{\partial x}\frac{dx}{ds} + \frac{\partial z}{\partial y}\frac{dy}{ds}$$

## Back propagation（BP神经网络）




我们知道神经网络中的损失函数为

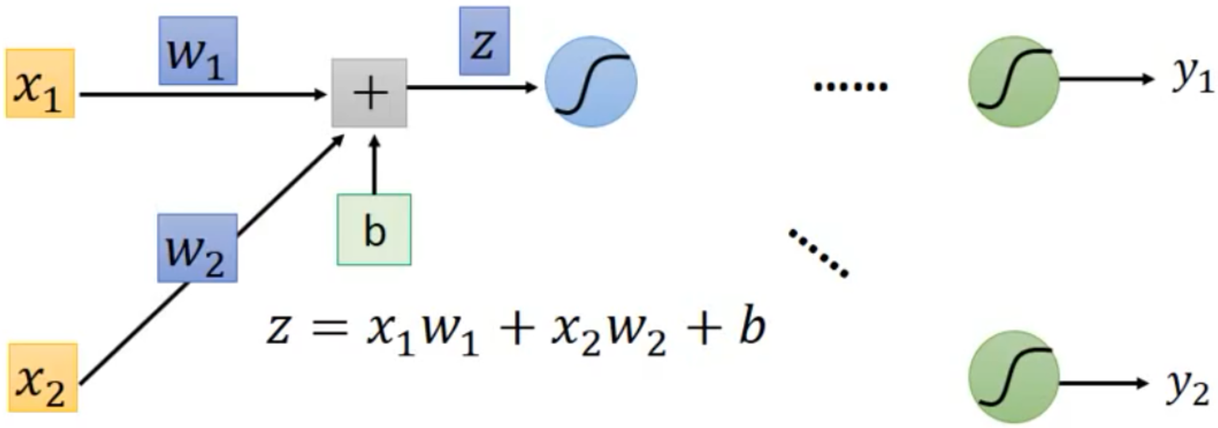
$$L(\theta) = \sum_{n=1}^{N} C^n(\theta)$$

我们对该损失函数同时进行某一个w的偏微分，则有

$$\frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{N} \frac{\partial C^n(\theta)}{\partial w}$$

这时我们只要将一个样本的偏微计算出来，然后遍历其他所有的样本，我们就可以把整个损失函数对某一项参数的偏微分计算出来。

## 计算方法




假设我们有两个样本输入到神经网络中，其中的权重为w1,w2，偏差值为b，则在第一层中，得到的结果z可以表示为

$$z = x_1w_1 + x_2w_2 + b$$

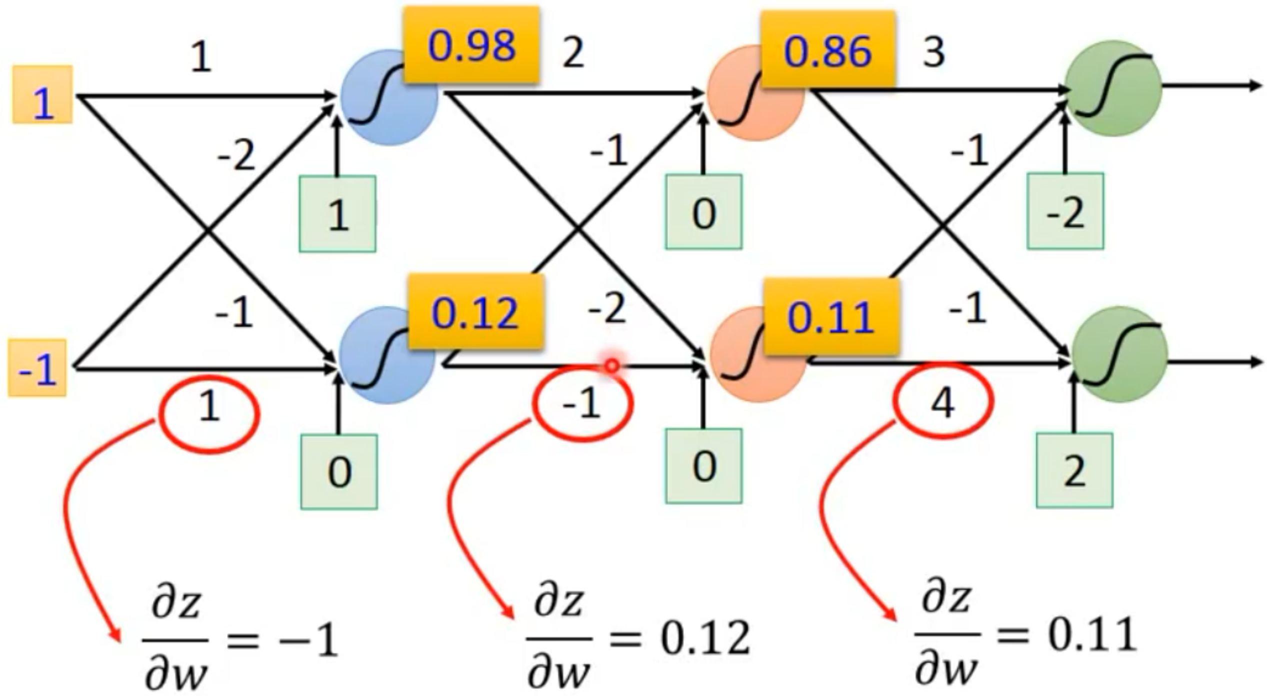
而在经过多层变化之后，输出y1,y2.而y1,y2与原有样本之间的交叉熵，我们可以视为C

**1.z对w的偏导**

这里，我们对w进行偏微分的推导过程：

$$\text{已知} z = x_1w_1 + x_2w_2 + b,$$
$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial z}\frac{\partial z}{\partial w}$$
$$\text{而} \frac{\partial z}{\partial w_1} = x_1,$$
$$\frac{\partial z}{\partial w_2} = x_2$$

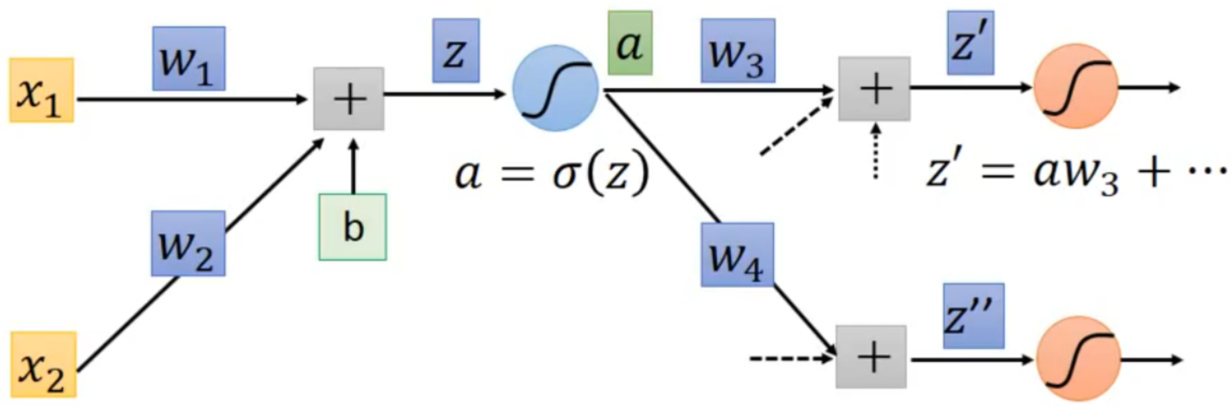
而进行多次推导，其偏导的值如下：




我们根据上面的公式可以看到，对w1和w2求偏导之后，其值都等于前一层的输入值。即等于x1,x2。

所以我们可以得出一个结论。即：**某一层对z求w的偏导的值，等于上一层的输入值。**

**2.C对z的偏导**




我们再回到一开始的神经网络的演化流程，我们已知每一层的神经元是sigmoid函数，那么我们可以假设神经元的sigmoid为

$$a = \sigma(z)$$

其中

$$z = x_1 w_1 + x_2 w_2 + b$$

在经过w3，w4的进一步变化之后得到了

$$z^{'} = aw_3 + \dots$$
$$z^{''} = aw_4 + \dots$$

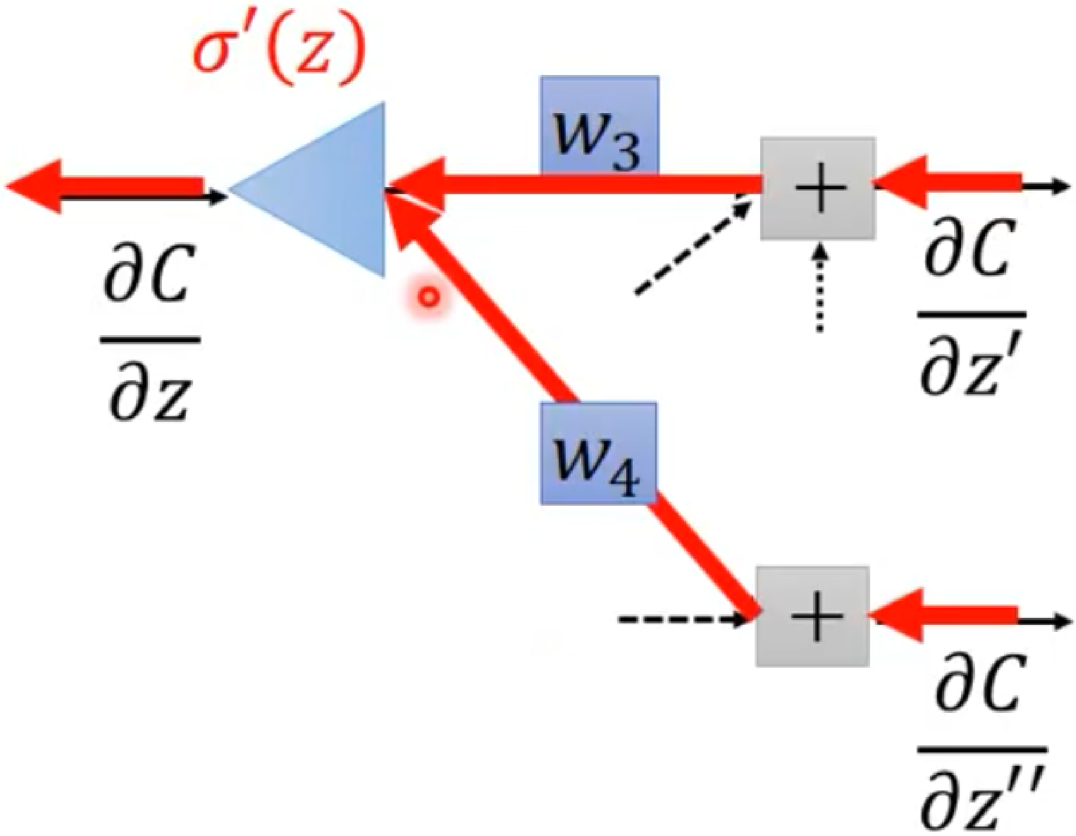
因此我们可以得到

$$\frac{\partial C}{\partial z}=\frac{\partial a}{\partial z}\frac{\partial C}{\partial a}$$

$$\text{其中}\frac{\partial C}{\partial a}=\frac{\partial z^{'}}{\partial a}\frac{\partial C}{\partial z^{'}}+\frac{\partial z^{''}}{\partial a}\frac{\partial C}{\partial z^{''}}$$

$$\text{而}\frac{\partial z^{'}}{\partial a}=w_3,\frac{\partial z^{''}}{\partial a}=w_4,$$

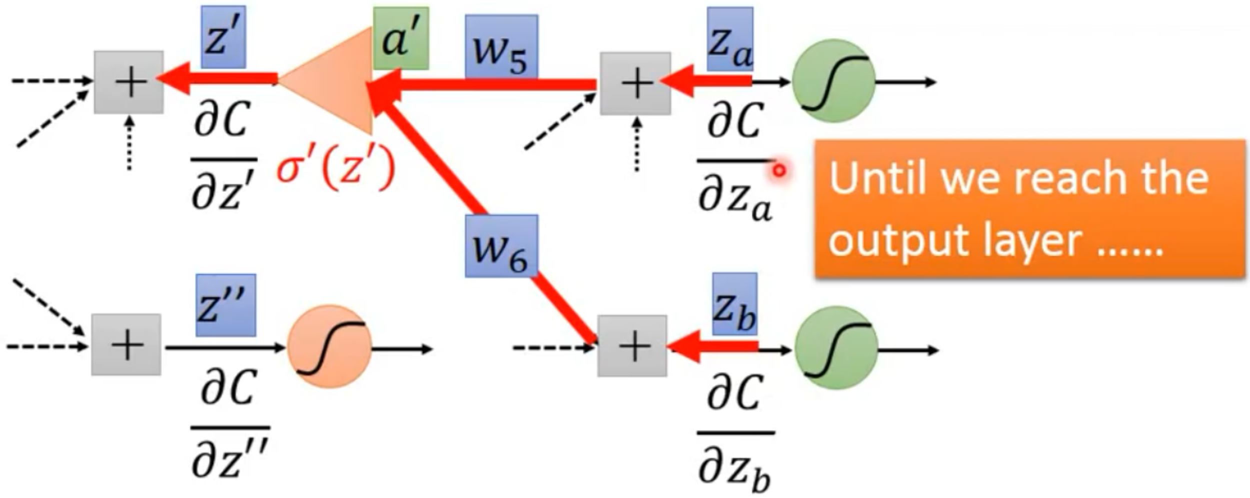
$$\text{因此我们可以得到}\frac{\partial C}{\partial z}=\sigma^{'}(z)\left[w_3\frac{\partial C}{\partial z^{'}}+w_4\frac{\partial C}{\partial z^{''}}\right]$$




这时候我们可以反着过来看，将整个推导顺序颠倒一下可以看成为一个反方向的神经网络。那么这个时候公式

$$\frac{\partial C}{\partial z}=\sigma^{'}(z)\left[w_3\frac{\partial C}{\partial z^{'}}+w_4\frac{\partial C}{\partial z^{''}}\right]$$

同样成立。并且由于z是已经早就已经决定好了的，所以z的导数也是一个常量。




因此依照类推，我们可以将后面所有层次的神经元进行相同操作，知道最后我们达到了输出层，则停止。从而由后面输出层的值，来回推出前面对于z的偏微分。

在前推中计算出了z对w的偏导，而后推中计算出的C对z的偏导，二者相乘，就可以得到C对w的偏导，从而完成了起初函数的梯度下降的演算过程。