1.3 Backpropagation

作用： compute the gradient efficiently
原理： 导数的链式法则

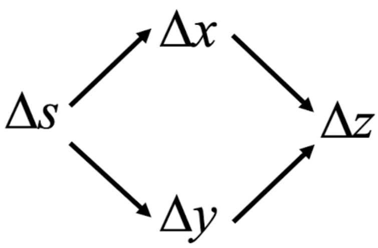
Case 1

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

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

Case 2

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



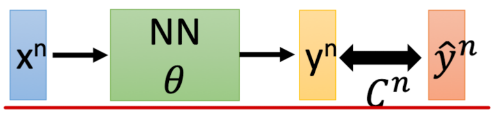
$$\frac{dz}{ds} = \frac{\partial z}{\partial x}\frac{dx}{ds} + \frac{\partial z}{\partial y}\frac{dy}{ds}$$

只关注某一笔data的偏微分

以weight为例，b同理 | 对每个y^n都有C^n

Backpropagation

定义y^与ŷ^距离的函数C^，越小越好.



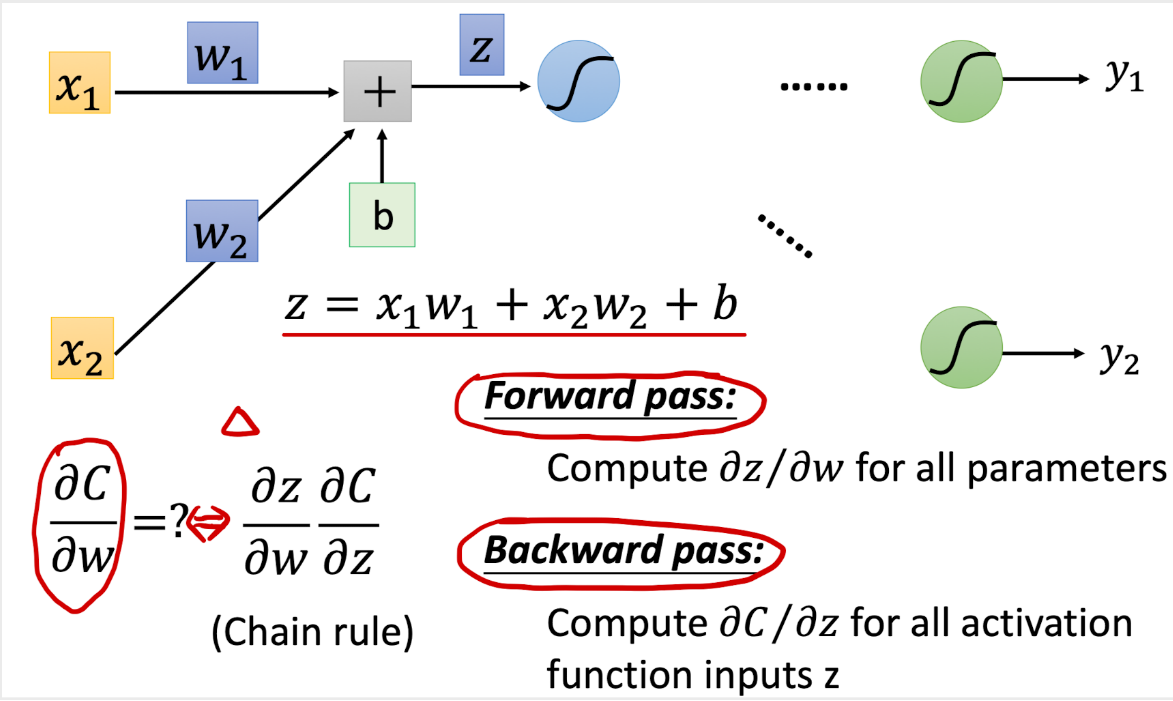
之前所有training data的某个loss值的和.

上标，不是n次方

$$L(\theta) = \sum_{n=1}^{N} C^n(\theta) \quad \xrightarrow{\text{对w偏微分}} \quad \frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{N} \boxed{\frac{\partial C^n(\theta)}{\partial w}}$$

只focus对每一笔data的偏微分

以下图的neural为例，C对w的偏微分可以用链式法则一拆分成两项，前项称作Froward pass，后项称作Backward pass。

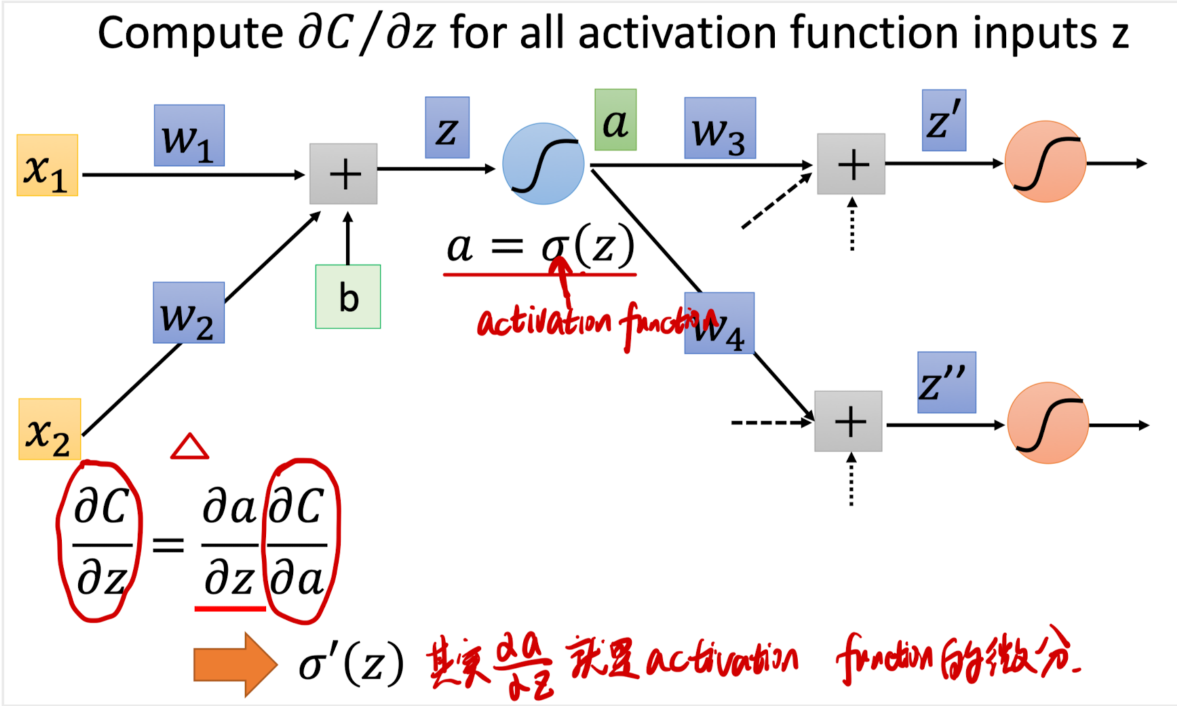


Forward pass

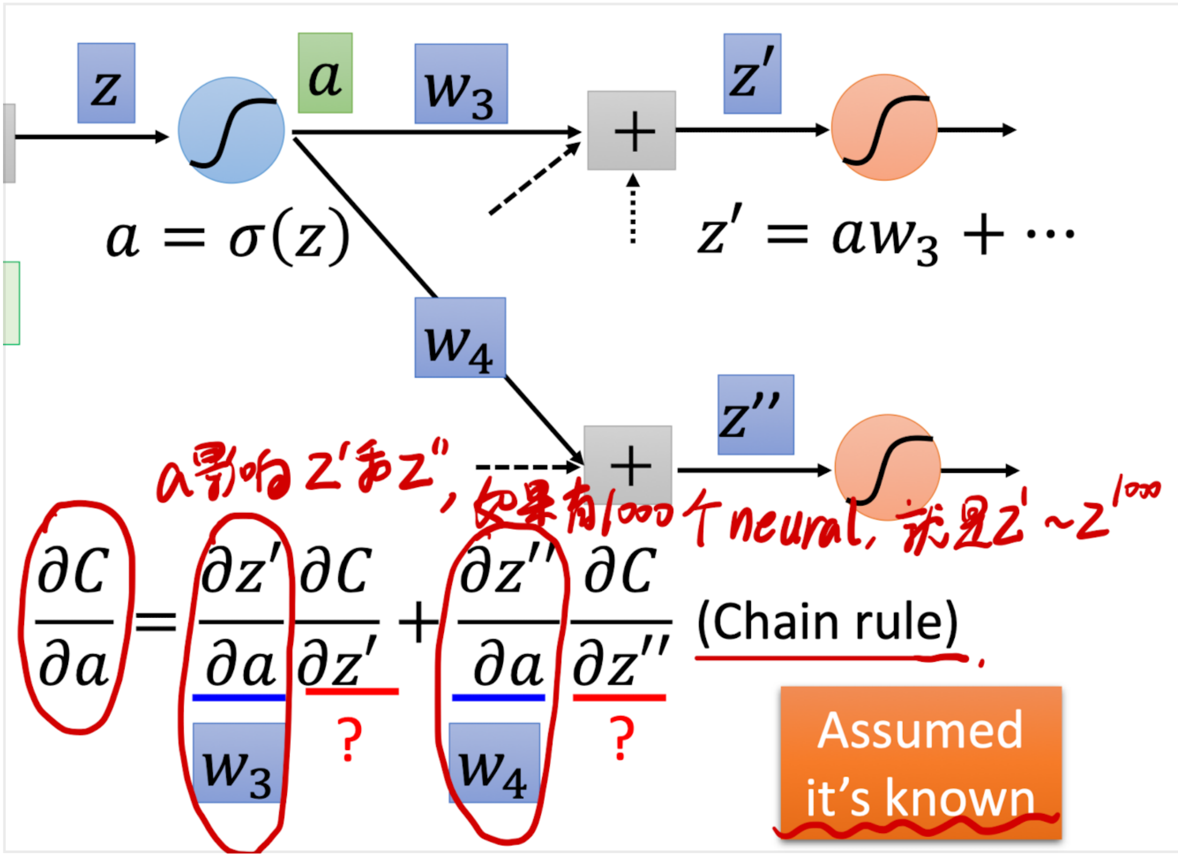
通过这样，可以算出每个参数的偏微分并记录。

Backward pass

再用链式法则一，假设 activation function 是 sigmoid function



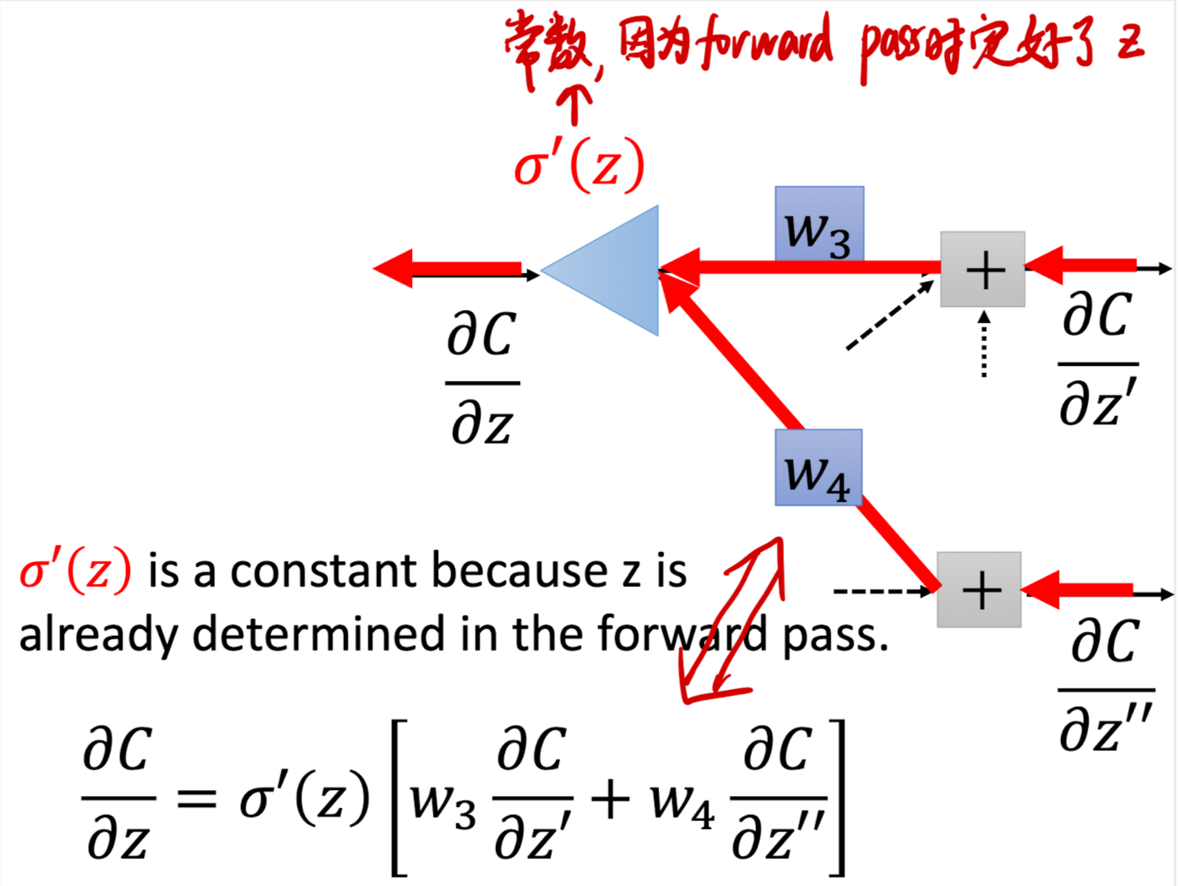
C对a的偏导数，用链式法则二，并假设C对z的偏导是知道的



假设知道 $\frac{\partial C}{\partial z'}$ 和 $\frac{\partial C}{\partial z''}$，则会算 $\frac{\partial C}{\partial z}$

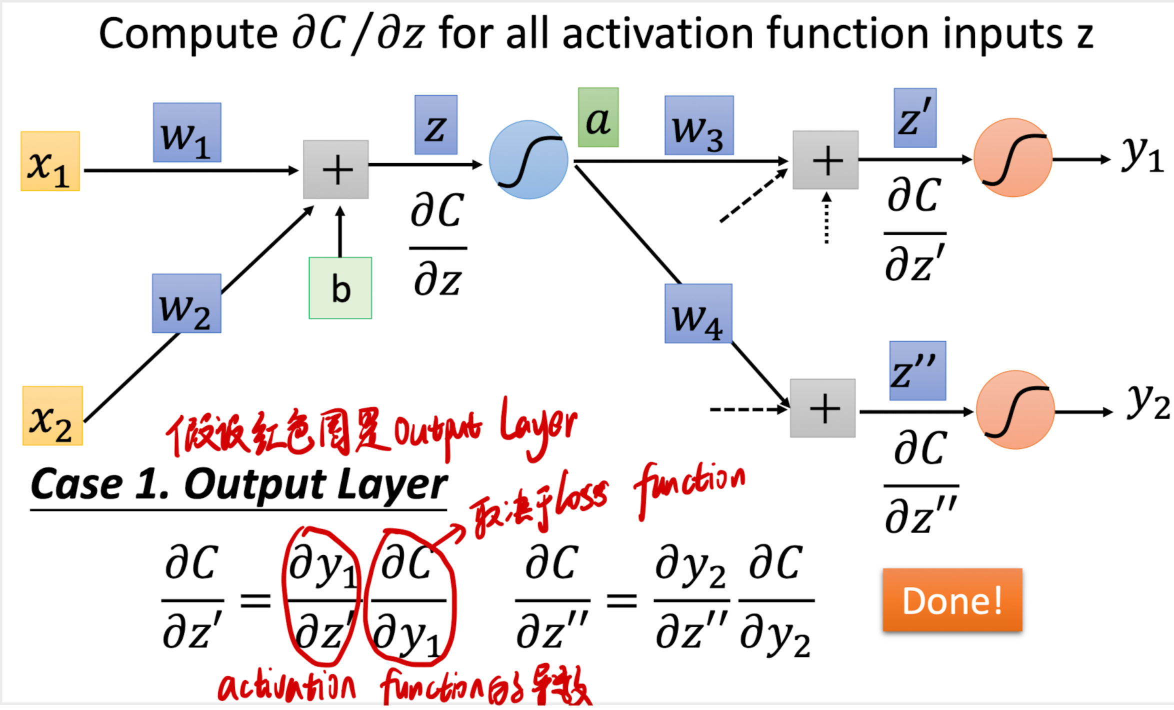
$$\frac{\partial C}{\partial z} = \sigma'(z)\left[w_3 \frac{\partial C}{\partial z'} + w_4 \frac{\partial C}{\partial z''}\right]$$

现在想象有另外一个neural，如下图三角形所示，input是C对z′和z′′的偏导，图中的neural做的事情和图中的公式是等价的。其中乘法左边一项是常数

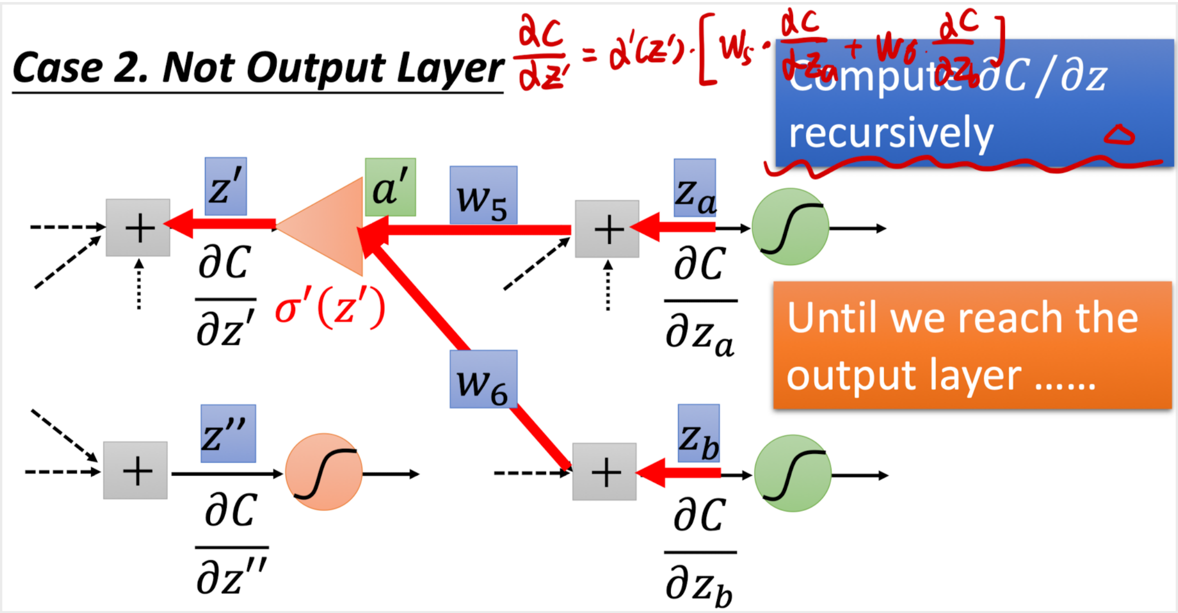


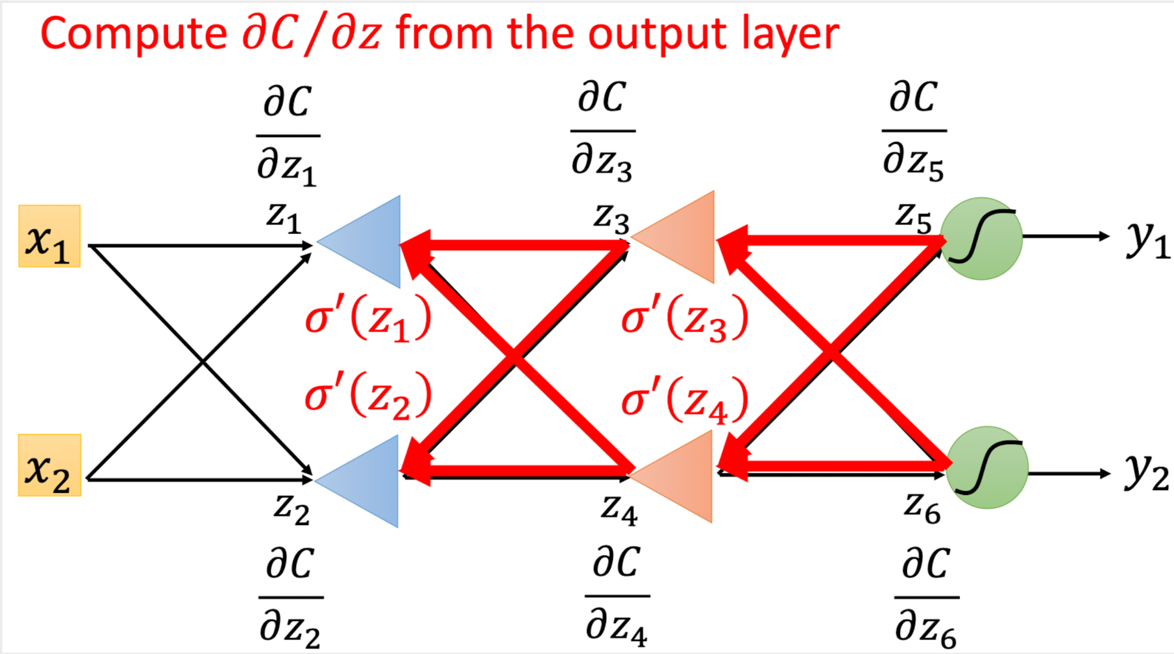
所以问题就在于如何算C对z′的偏导和C对z′′的偏导。分两种情况讨论；

Case1:假设红色neural是output layer

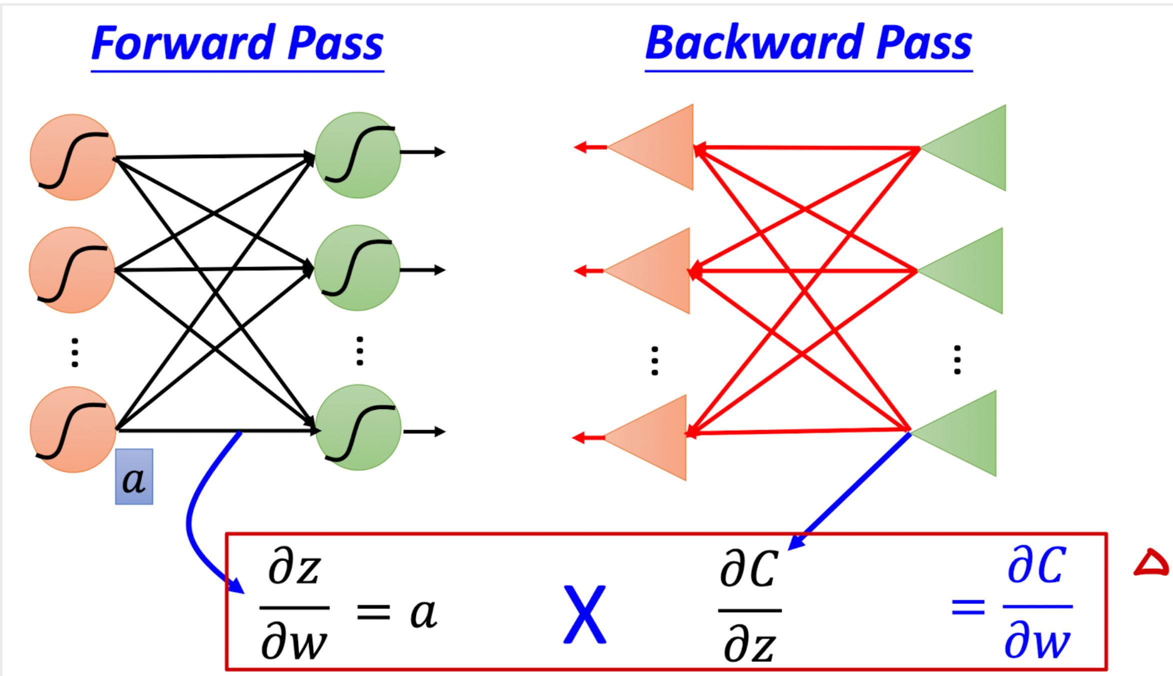


Case2:假设红色圈后面仍有 layer，则计算输出层的 C 对 z 的偏导，然后往前推





Summary



forward pass 可以算出每个 activation 的 output (即 $\frac{\partial z}{\partial w}$)