Backpropagation

在 Neural Network 里，要用 Gradient Descent 来 Update 参数的步骤就叫做 Backpropagation。在 Neural Network 里，参数太多 (Millions of Parameters)，所以$\nabla L(\theta)$ 的 Vector Size 是非常大的。Backpropagation 能有更有效的 Update 这些参数。

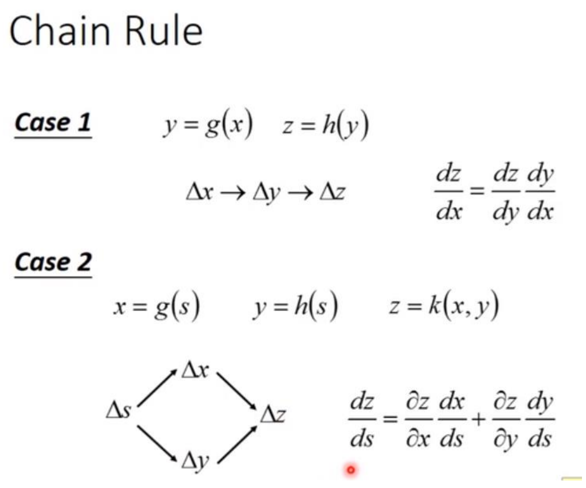
$$Network\ Parameters\ \theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots\}$$

$$Gradient\ Descent \rightarrow\ \theta^{n+1} = \theta^n - \eta\nabla L(\theta^n)$$

$$\nabla L(\theta) = \begin{bmatrix} \dfrac{\partial L(\theta)}{\partial w_1} \\ \dfrac{\partial L(\theta)}{\partial w_2} \\ \vdots \\ \dfrac{\partial L(\theta)}{\partial b_1} \\ \dfrac{\partial L(\theta)}{\partial b_2} \\ \vdots \end{bmatrix}$$

这个 Process 是持续进行的，计算 Forward Propagation，然后用算出来的 $output\ y$ 计算 Loss Function，然后计算$\nabla L(\theta^n)$，然后计算 Gradient Descent 来 Update 参数。

Chain Rule



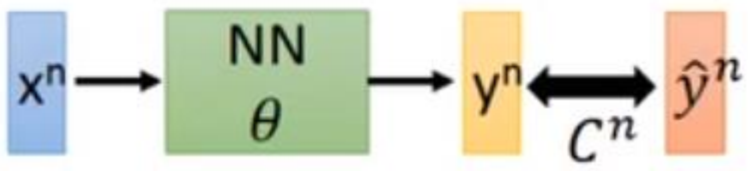
看 Case1，当 x 有变化的时候，就会影响到 y，而 y 有了变化，就会影响到 z。做 Chain Rule 其实就是要找$\frac{dz}{dx}$，当 x 有变化后，会影响 z 多少。

看 Case2，当改变 s 的时候，同时会改变 x 和 y，而 x 和 y 改变后会影响到 z。s 对 z 的微分就是：

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

Backpropagation 计算方法

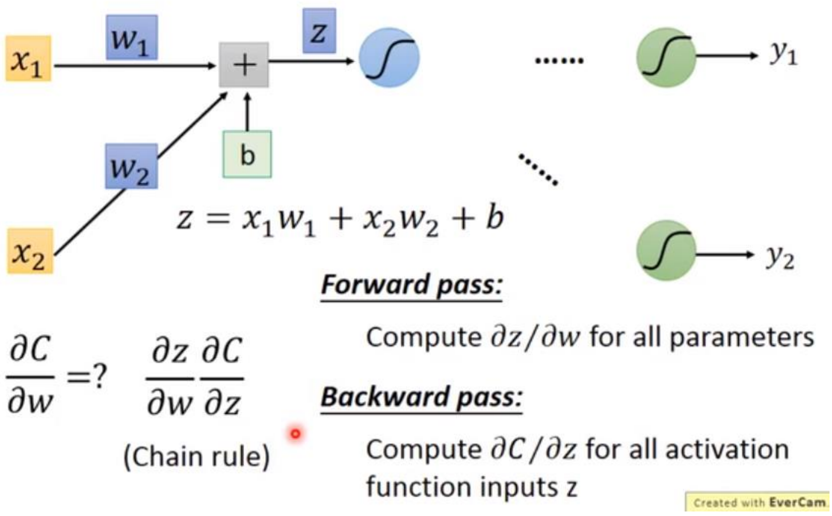
$$L(\theta)=\sum_{n=1}^{N}C^{n}(\theta)$$



在训练模型之前，需要定义一个 Loss Function。这个 Loss Function 可以是 MSE，Cross Entropy 又或者是其他的 Loss Function。

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

做了偏微分就能得到上面的这个 Equation。



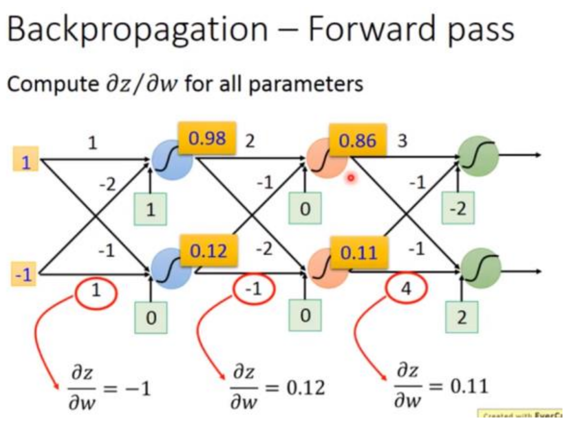
$$z=x_1w_1+x_2w_2+b$$

$$Chain\ Rule\rightarrow\frac{\partial C}{\partial w}=\frac{\partial z}{\partial w}\frac{\partial C}{\partial z}$$

在执行 Backpropagation 的时候，我们需要 Update 每一个在 Neural Network 里面的参数$weight$和 $bias$。在这里解释只争对$weight$，而$bias$的做法其实就和$weight$ 一样。要 Update 这个 $weight$，就需要算出$\frac{\partial C}{\partial w}$，就是这个 $weight$ 会影响 Loss 多少。

$$\frac{\partial z}{\partial w_1}=x_1$$

$\frac{\partial z}{\partial w}$ 的规律就是在 $weight$ 的前面接的是什么，算出来的微分就是前面接的值。w_1 前面接受的是 x_1，算出来的$\frac{\partial z}{\partial w_1}$ 就是x_1。

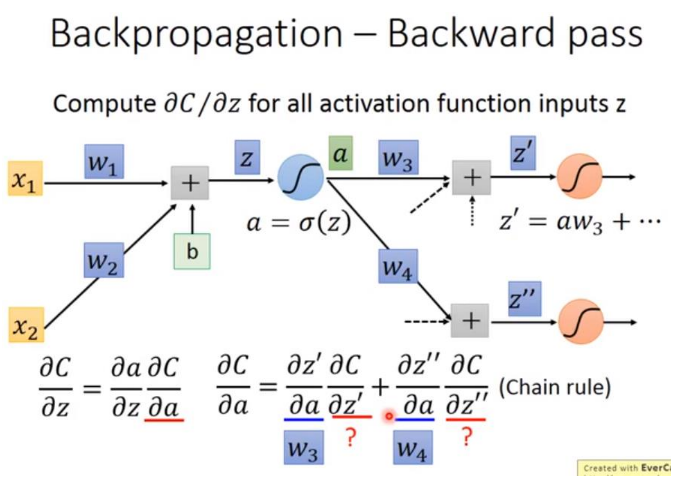


在训练一个 Neural Network，都需要 Update 每一层的参数，也就是找每一个参数的$\frac{\partial z}{\partial w}$。这时候就是 Forward Propagation 在做的事情。每一个 Neuron 的 Output 其实就是 $\frac{\partial z}{\partial w}$。

$$z = x_1w_1 + x_2w_2 + b$$

$$\boldsymbol{Chain\ Rule \rightarrow \frac{\partial C}{\partial w} = \frac{\partial z}{\partial w}\frac{\partial C}{\partial z}}$$

当 $\frac{\partial C}{\partial w}$ 已经有办法计算出来后，这时候需要计算的就是$\frac{\partial C}{\partial z}$。要计算 $\frac{\partial C}{\partial z}$ 的话需要在进行多一次的 Chain Rule。在这里，使用的 Activation Function 是 Sigmoid Function。如果是其他的 Activation Function，那么计算的 Partial Derivation 需要争对该 Activation Function 来计算。



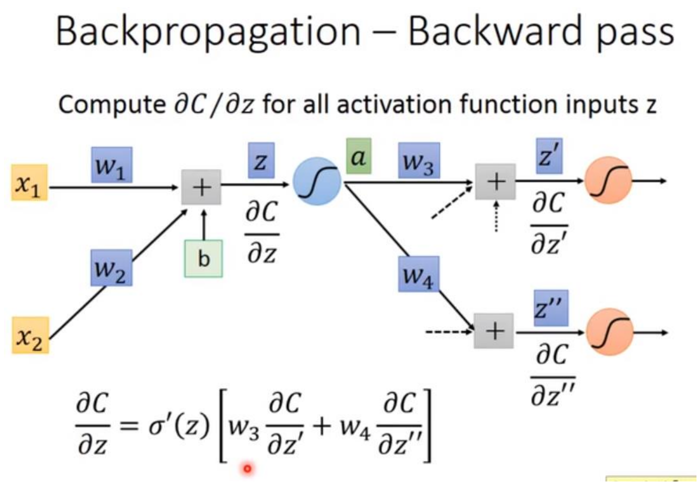
$$a = \sigma(z) \rightarrow a\ is\ Output\ of\ Sigmoid\ Function$$

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

$$\frac{\partial a}{\partial z} = \sigma'^{(z)} \rightarrow Constant\ Value$$

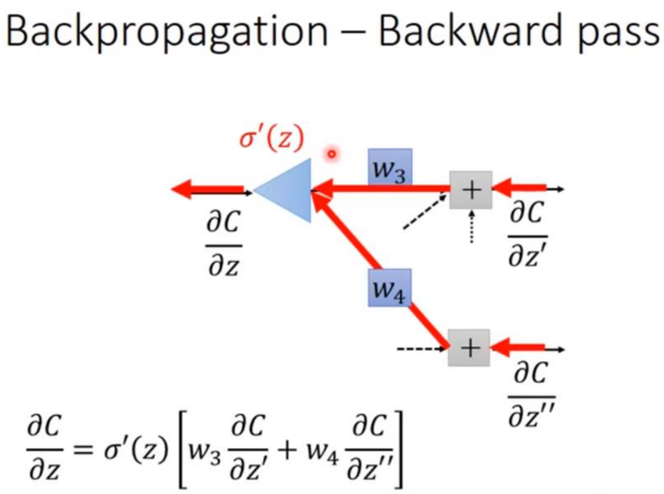
再上图，a 是会影响 z’ 而 z’ 会影响接下来的 Layer。如果同一个 Layer 有 100 个 Neuron 那么$\frac{\partial C}{\partial a}$的 Summation 就需要是这 100 个 Neuron。

$$\boldsymbol{\frac{\partial C}{\partial a} = \frac{\partial z'}{\partial a}\frac{\partial C}{\partial z'} + \frac{\partial z''}{\partial a}\frac{\partial C}{\partial z''}}$$



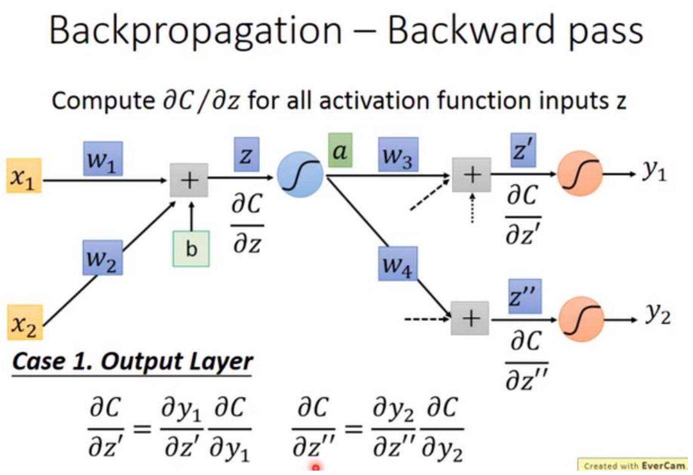
当计算出了下一个 Layer 的 $\frac{\partial C}{\partial z'}$ 和 $\frac{\partial C}{\partial z''}$，就能找出 $\frac{\partial C}{\partial z}$。

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



推算到这里就能发现，其实就是把 Neural Network 反着算。算出当前层的 Partial Derivative 就能 Update 前一层的参数。

所以做 Backpropagation 的第一个步骤就是先做 Forward Propagation 然后把每一个 Neuron 的 Output 都先记录下来。然后执行 Backpropagation 的时候就是从 Neural Network 的 Output Layer 开始做。



$$z = x_1 w_1 + b \text{ and } z' = a w_3 + b$$

$$y = \sigma(z)$$

$$Cross\ Entropy\ Loss \rightarrow \sum_n -[\hat{y}^n \ln y^n + (1-\hat{y}^n)\ln(1-y^n)]$$

首先要找出 w_3 对 Loss Function 的影响 $\frac{\partial C}{\partial w_3}$

$$\frac{\partial C}{\partial w_3} = \frac{\partial C}{\partial z'}\frac{\partial z'}{\partial w_3}$$

$$\frac{\partial z'}{\partial w_3} = a$$

没办法直接找$\frac{\partial C}{\partial z'}$ 所以要做多一次 Chain Rule

$$\frac{\partial C}{\partial z'} = \frac{\partial C}{\partial y_1}\frac{\partial y_1}{\partial z'}$$

$$\frac{\partial y_1}{\partial z'} = \sigma'(z') = \frac{e^{-z'}}{(e^{-z'}+1)^2}$$

$$\frac{\partial C}{\partial y_1} = \sum_n -(\hat{y}^n - y^n) = \sum_n y^n - \hat{y}^n$$

$$\frac{\partial C}{\partial z'} = \sum_n y^n - \hat{y}^n * \frac{e^{-z'}}{(e^{-z'}+1)^2}$$

找到 $\frac{\partial C}{\partial z'}$ 之后就能找到 $\frac{\partial C}{\partial w_3}$

$$\frac{\partial C}{\partial w_3} = \sum_n y^n - \hat{y}^n * \frac{e^{-z'}}{\left(e^{-z'}+1\right)^2} * a$$

算出了$\frac{\partial C}{\partial w_3}$之后，就能计算出$\frac{\partial C}{\partial w_1}$

$$\boldsymbol{\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial z}\frac{\partial z}{\partial w_1}}$$

$$\frac{\partial z}{\partial w_1} = x_1$$

没办法直接找$\frac{\partial C}{\partial z}$所以要做多一次 Chain Rule

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

$$\frac{\partial a}{\partial z} = \sigma'(z) = \frac{e^{-z}}{(e^{-z}+1)^2}$$

没办法直接找$\frac{\partial C}{\partial a}$所以要做多一次 Chain Rule

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

$$\frac{\partial z'}{\partial a} = w_3$$

$$\frac{\partial z''}{\partial a} = w_4$$

$$\textit{之前计算过} \rightarrow \boldsymbol{\frac{\partial C}{\partial z'} = \sum_n y^n - \hat{y}^n * \frac{e^{-z'}}{(e^{-z'}+1)^2}}$$

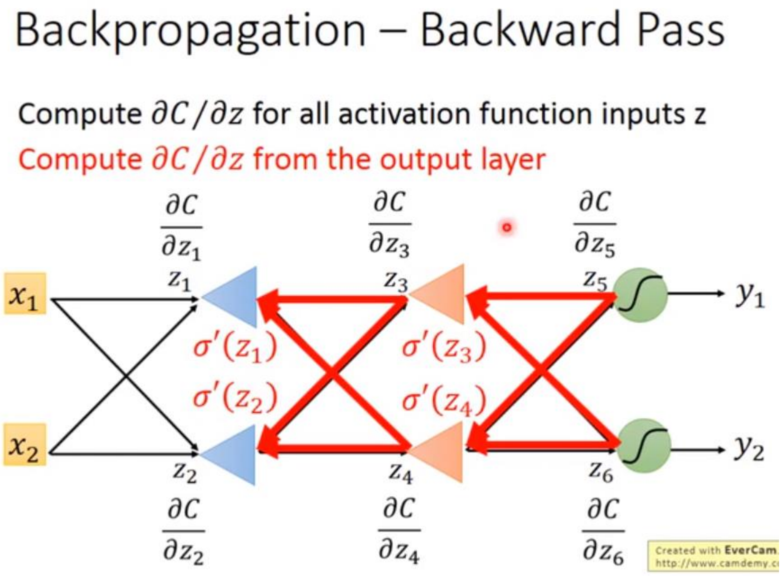
找到$\frac{\partial C}{\partial a}$之后就能找到$\frac{\partial C}{\partial z}$

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

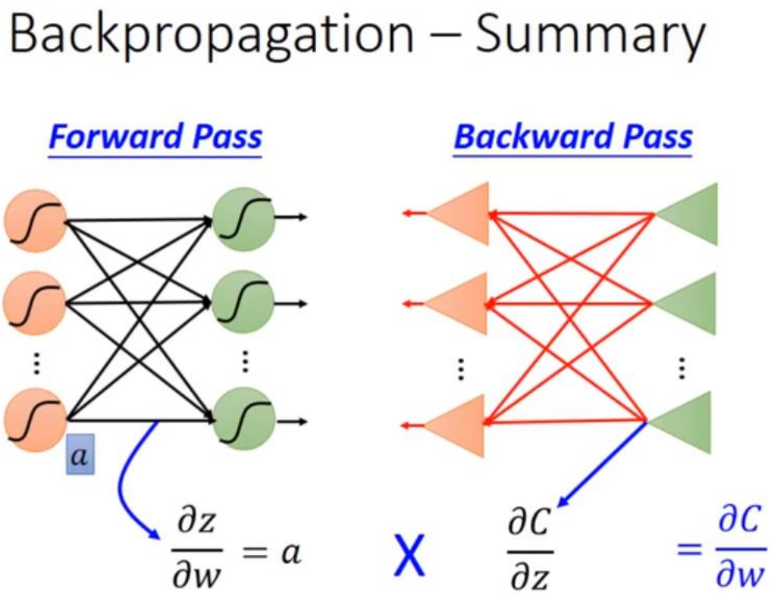
找到$\frac{\partial C}{\partial z}$之后就能找到$\frac{\partial C}{\partial w_1}$

$$\boldsymbol{\frac{\partial C}{\partial w_1} = \sigma'(z)\left(w_3\frac{\partial C}{\partial z'} + w_4\frac{\partial C}{\partial z''}\right) * x_1}$$

Summary



只要先从 Neural Network 的 Output 开始计算每一个 neuron 的 Partial Derivative 就能轻松执行 Backpropagation。



Backpropagation 就是先做 Forward Pass 来计算出 a 然后做 Backward Pass 计算出 Partial Derivative 就能计算出 $\frac{\partial C}{\partial w}$，之后就能放入 Gradient Descent 的 Function 来 Update 参数。