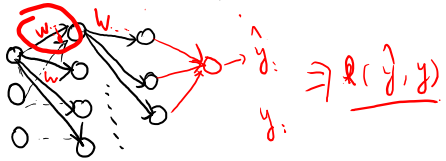
反向传播之即算法的推导：

1. 为什么会有反向传播算法产生：先看图：



$\Rightarrow l(\hat{y}, y)$

现：把求得参数（权重 matrix）的过程更新，就应该清楚参数关于输出的梯度。

但是：请大家观察一下，参数关于 output 的 gradient 可以直接一步到位的计算出来吗？

答案显然是不可以的！

反向传播之前定义 N 个符号以便理解：

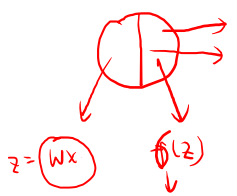
1°、这第 l 层的参数为 $W^{(l)}$

$W_{ij}^{(l)}$ 表示：第 l 层的第 i 个神经元与第 $(l+1)$ 层的第 j 个神经元的连接权重！

2°、$l(y, \hat{y})$ 表示输出层的损失函数 loss.

3°、记：$\begin{cases} z^{(l)} = W^{(l)} a^{(l-1)} \\ z^{(l+1)} = W^{(l+1)} a^{(l)} \end{cases}$

$\begin{cases} a^{(l)} = \sigma(z^{(l)}) \\ a^{(l+1)} = \sigma(z^{(l+1)}) \end{cases}$



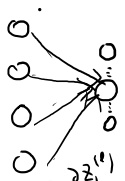
其中：$\sigma(\cdot)$ 为 sigmoid 函数.

$$\frac{\partial l(y, \hat{y})}{\partial w_{ij}^{(l)}} = \frac{\partial l(y, \hat{y})}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$$

第二项这项怎么得的？

关键：$\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$ 怎么求

$\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$



有点：大家想清楚，z 是什么？

$z = [z_1, z_2, \cdots, z_N]$

$$\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}} = \left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \frac{\partial z_2^{(l)}}{\partial w_{ij}^{(l)}}, \cdots \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \cdots\right]$$

$$= \left[0, 0, \cdots \frac{\partial (w_{i:}^{(l)} \cdot a^{(l-1)})}{\partial w_{ij}}, 0, \cdots\right]$$

$$= [0, 0, \cdots a_j^{(l-1)}, \cdots 0, 0]$$

$$= \mathbb{V} a_j^{(l-1)}$$

$z^{(l)} = w^{(l)} \cdot a^{(l-1)}$

再拿着：$\frac{\partial l(y,\hat{y})}{\partial z^{(l)}}$ ：记：$\frac{\partial l(y,\hat{y})}{\partial z^{(l)}} \triangleq \delta^{(l)}$

$$\frac{\partial l(y,\hat{y})}{\partial z^{(l)}} = \frac{\partial l(y,\hat{y})}{\partial z^{(l+1)}} \cdot \frac{\partial z^{(l+1)}}{\partial z^{(l)}}$$

$$= \frac{\partial l(y,\hat{y})}{\partial z^{(l+1)}} \cdot \frac{\partial z^{(l+1)}}{\partial a^{(l)}} \cdot \frac{\partial a^{(l)}}{\partial z^{(l)}}$$

$\delta^{(l)}$ ⇓ $\delta^{(l+1)}$ ⇓ $w^{(l+1)}$ ⇓ $\sigma'(\cdot)$

$a^{(l)} = \sigma(z^{(l)})$

$z^{(l+1)} = w^{(l+1)} \cdot a^{(l)}$

$z^{(l)} = [z_1^{(l)}, \cdots z_N^{(l)}]$

$\sigma(1-\sigma)$

$\sigma(z^{(l)})$

故：$\frac{\partial l(y,\hat{y})}{\partial w_{ij}^{(l)}} = \mathbb{V} a_j^{(l-1)} \cdot \delta^{(l+1)} \cdot w^{(l+1)} \cdot \sigma'(\cdot)$

$= \mathbb{V} a_j^{(l-1)} \cdot \delta^{(l)}$

矩阵形式：$\delta^{(l)} (a^{(l-1)})^T$

$\delta^{(l)} = \delta^{(l+1)} \cdot w^{(l+1)} \cdot \sigma'(\cdot)$

$\delta^{(l)}$ $\delta^{(l+1)}$ → output

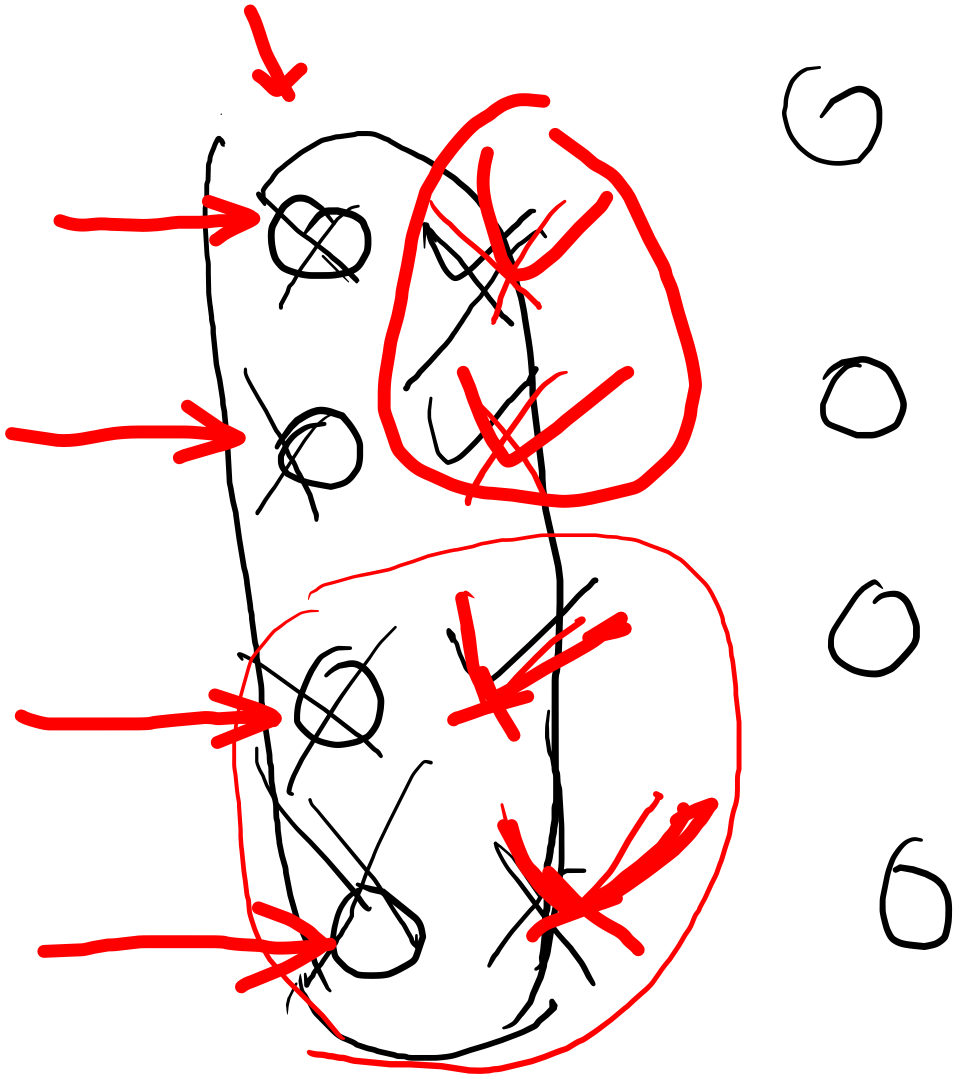
$\delta^{(output)}$

recursion

$\mathbb{V} a^{(l-1)}$ $(z^{(l)})$ ⇓ $a^{(l)}$

1. ~~删~~ dropout

2. BN : Batch-Normalization



0.5

×0.5

缩放的 × 手动

加合信息的能力 ↗

1、dropout 不是分段的

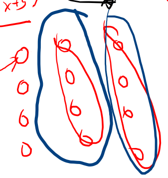
$y = y_1 + y_2 \to Wx \cdot W^2x$

$y' = y_1 \cdot y_2$

2、BN 与 dropout ⇓ 不调置

$\sigma(Wx+b)$ y $x^2 \cdot W^3$

μ, σ^2 $y_1 = Wx$ $y_2 = W^2x$



Batch_Norma

⇓

μ, σ^2

$\{x_1, x_2, \ldots x_j, \ldots x_{B\_size}\}$

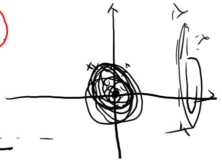
⇓

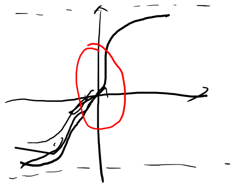
μ, σ^2

$$\mu = \frac{1}{N}\sum_{j=1}^{N} x_j$$

$$\sigma^2 = \frac{1}{N}\sum_{j=1}^{N} (x_j - \mu)^2$$

$$\frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}$$





tanh(·)

⇓

(-1, 1)

1、梯度（缓解）
2、训练速度快 ↑
3、初始化方式
4、learning_rate 可稍大

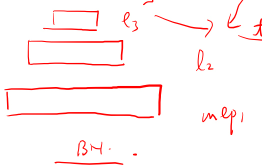
NAN → lr ↓ √

BN

$$\frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}$$

$$y_i = a\frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + b$$

trainable



RNN、LSTM ✗

⇓

50% 反效果

Layer_Norm

Sequence μ, σ^2