Coss function of Neural Network

定义：{ L：网络的层数
S_l：第 l 层的单元个数；
K：输出层里单元的个数

$h_\Theta(x)_k$ 代表输出的第 k 个类别的预测值

回顾：logistic regression 中的 cost function（无双和）

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

类比，神经网络用在多分类时的 loss，就是增加了一些数值求和。

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log(h_\Theta(x^{(i)})_k) + (1-y_k^{(i)})\log(1-h_\Theta(x^{(i)})_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

（PS：用 sigmoid 函数将输出映射成概率，值 $\in [0,1]$，且用于多分类时，其总和不一定是 1；softmax 是 1）

类之间不排斥，可能有多个正确答案。

第一部分：对 output layer 的每个 cell 都计算 logistic loss 并求和（$\sum\sum$）

before layer ... last layer

第二部分：对每个 Θ 参数算平方并求和（$\sum\sum\sum$）

layer

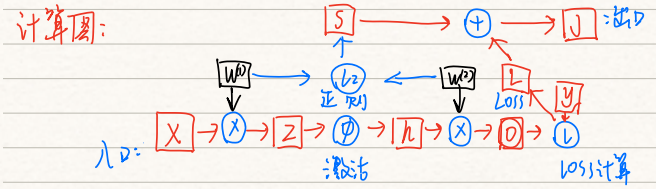
Backpropagation Algorithm

以前只知道"反向传播"这个词，不知道它是一种算法

反向传播和 GD 用在线性回归中一样，目标是最小化损失函数。也是一种参数优化（寻找到一组最合适的 θ_{ij}）方法。 ——逻辑→ 指的是计算神经网络参数梯度的方法

Forward propagation：正向传播（FP）：按顺序（从输入层到输出层）计算和存储神经网络中每层的结果

计算图：



Backpropagation（反向传播，BP）：根据微积分中链式法则的规则，从输出层到输入层的顺序

通过一次正向传播，得到了预测值。

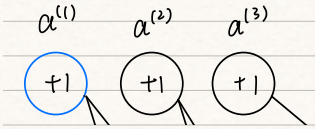
通过一次 反向传播，更新了参数（一组 weight），使得再一次 FP 得到的输出与真实值之间的误差减小。

注意："前向"：从 input 层到 output 层

"前一层"：箭头指向的前一层

反向传播：计算梯度，梯度就是误差下降最快的方向。

参数关于损失函数



FP：

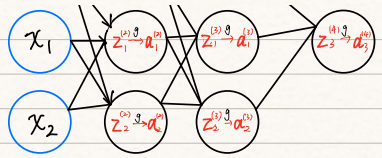
$a^{(1)} = x$

$z^{(2)} = \Theta^{(1)}a^{(1)}$

BP：从 output 层开始，计算 $\delta^{(l)}$

$\delta^{(4)} = a^{(4)} - y$（即预测值与真实值之差）

再利用 $\delta^{(l)} = ((\Theta^{(l)})^T\delta^{(l+1)}) .* a^{(l)} .* (1-a^{(l)})$

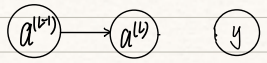


$$\begin{cases} a^{(2)} = g(z^{(2)}) \\ z^{(3)} = \Theta^{(2)} a^{(2)} \\ a^{(3)} = g(z^{(3)}) \\ z^{(4)} = \Theta^{(3)} a^{(3)} \\ a^{(4)} = g(z^{(4)}) = h_\Theta(x) \end{cases}$$

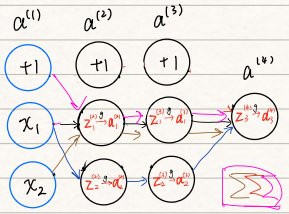
当层参数 后一层的 delta 项 即 $g'(z^{(l)})$

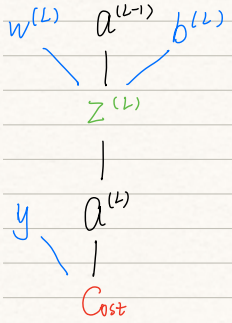
与链式法则联系：

假设每层仅有一个节点：



每层有多个结点，就是有很多条路，加求和符号！





计算 cost 对 w 的微小变化有多敏感：

$$\frac{\partial Cost}{\partial w^{(l)}} = \frac{\partial Cost}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial w^{(l)}}$$

最后一层　上一层　上一层