

深层神经网络的记号：

L : 神经网络的层数. $L=4$

$n^{[l]}$ 表示第 l 层的节点数. $n^{[1]}=5, n^{[2]}=5, n^{[3]}=3, n^{[4]}=n^{[L]}=1$; $n^{[0]}=n_x=3$

$a^{[l]}$ 表示第 l 层的激活函数, $a^{[l]}=g^{[l]}(z^{[l]})$; $a^{[0]}=x, a^{[L]}=\hat{y}$

$w^{[l]}, b^{[l]}$

1. 深层网络中的前向传播

(1) 对一个单独的训练样本 x

第1层: $z^{[0]} = W^{[0]}x + b^{[0]}, a^{[0]} = g^{[0]}(z^{[0]})$

第2层: $z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}, a^{[1]} = g^{[1]}(z^{[1]})$

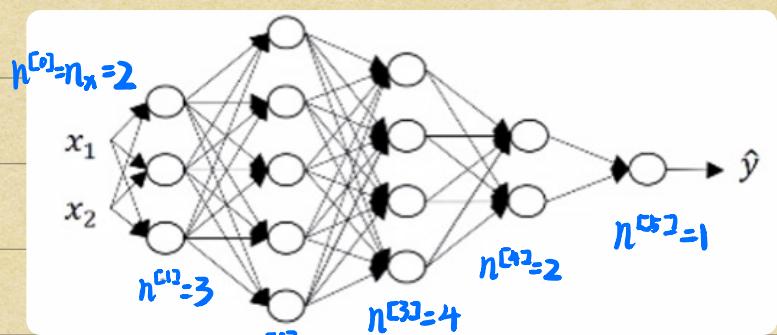
....

(2) 向量化:

for l in NN-layers: // 这里在遍历各个层时，不好避免的一个 for 循环

$z^{[l]} = W^{[l]}A^{[l]} + b^{[l]}, A^{[l]} = g^{[l]}(z^{[l]})$

2. 矩阵维度检查



(1) 一个样本时:

$$\text{维度分析: } \underbrace{\mathbf{z}^{[0]}}_{(n^{[0]}, 1)} = \underbrace{W^{[0]} \cdot \mathbf{x}}_{(n^{[0]}, n^{[1]})} + \underbrace{b^{[0]}}_{(n^{[1]}, 1)}$$

$$\mathbf{z}^{[1]} = W^{[1]} \cdot \mathbf{A}^{[0]} + b^{[1]}$$

规律

$$\left\{ \begin{array}{l} W^{[l]} : (n^{[l]}, n^{[l-1]}) \\ \mathbf{z}^{[l]} : (n^{[l]}, 1) \\ A^{[l]} : (n^{[l]}, 1) \\ b^{[l]} : (n^{[l]}, 1) \end{array} \right. \quad \begin{array}{l} \leftarrow d\mathbf{w} \text{ 与 } \mathbf{w} \text{ 维度相同} \\ \leftarrow \text{维度相同, } \mathbf{z} = g(\mathbf{A}) \end{array}$$

(2) m 个样本时:

$$\mathbf{z}^{[0]} = \begin{bmatrix} \mathbf{z}^{[0](1)} & \mathbf{z}^{[0](2)} & \dots \end{bmatrix} : (n^{[0]}, m), \quad W \text{ 仍为 } (n^{[0]}, n^{[1]})$$

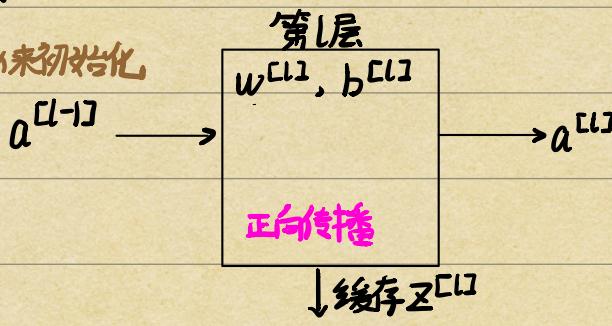
$$\mathbf{X} : (n^{[0]}, m)$$

3. 前向函数/反向函数

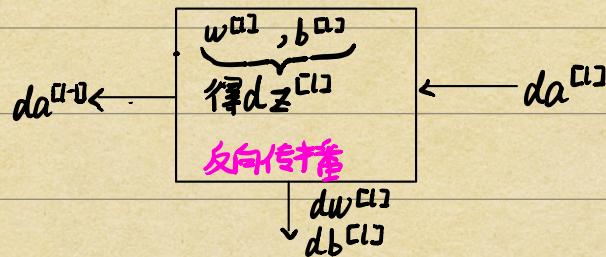
对第 l 层: $\begin{cases} \text{前向函数: 输入 } a^{[l-1]}, \text{ 输出 } a^{[l]}, \text{ 同时缓存 } \mathbf{z}^{[l]} \text{ 用于反向函数} \\ \text{反向函数: 输入 } da^{[l]}, \text{ 缓存的 } \mathbf{z}^{[l]}, \text{ 输出 } da^{[l-1]} \end{cases}$

→ 搭建神经网络块

→ 前向传播, 用 a 来初始化



正向传播时计算的参数
 $W^{[l]}, b^{[l]}, \mathbf{z}^{[l]}$
都可以缓存给反向传播使用.



→ 后向传播, 当是 Logistic 二元分类时, 初始化:
 $da^{[l]} = -\frac{y}{a} + \frac{(1-y)}{1-a}$

最末层

$$\Rightarrow \text{得到 } dw^{[l]}, db^{[l]}, \text{ 即可梯度下降} \begin{cases} W^{[l]} = W^{[l]} - \alpha dw^{[l]} \\ b^{[l]} = b^{[l]} - \alpha db^{[l]} \end{cases}$$

前向传播:

$$\begin{cases} z^{[l]} = W^{[l]}X + b^{[l]} & , A^{[l]} = g(z^{[l]}) \\ z^{[l]} = W^{[l]} \cdot X + b^{[l]} & , A^{[l]} = g(z^{[l]}) \end{cases} \quad \begin{matrix} \nearrow A \text{ 与 } z \text{ 维度相同} \\ \nearrow \text{元素对应相乘} \end{matrix}$$

反向传播:

$$dz^{[l]} = da^{[l]} \cdot \underbrace{g^{[l]'}(z^{[l]})}_{\text{激活函数的导数}} \quad \text{链式法则: } dz = da \cdot g'(z)$$

$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]T}$$

$$db^{[l]} = dz^{[l]}$$

$$da^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

注: $dz^{[l]} = W^{[l]T} \cdot dz^{[l-1]} \cdot g^{[l]'}(z^{[l]})$

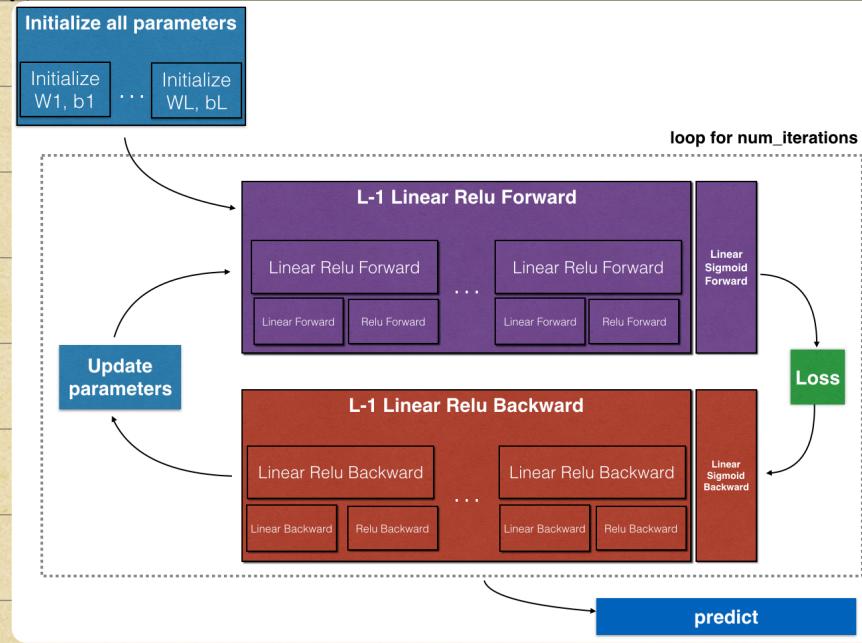
向量化:

$$\begin{cases} dz^{[l]} = da^{[l]} \cdot \underbrace{g^{[l]'}(z^{[l]})}_{\text{激活函数的导数}} \\ dw^{[l]} = \frac{1}{m} \cdot dz^{[l]} \cdot A^{[l-1]T} \\ db^{[l]} = \frac{1}{m} \cdot \text{np.sum}(dz^{[l]}, axis=1, keepdims=True) \\ dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]} \end{cases}$$

超参数 Hyperparameters:

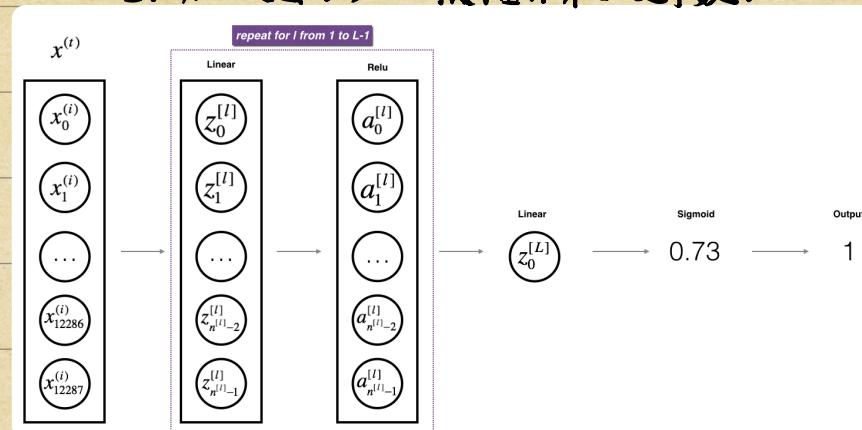
- 学习率 α
- 迭代次数
- 隐藏层数 L
- 隐藏单元数 $n^{[1]}, n^{[2]}, \dots$
- 激活函数
- ...

深度学习过程简介

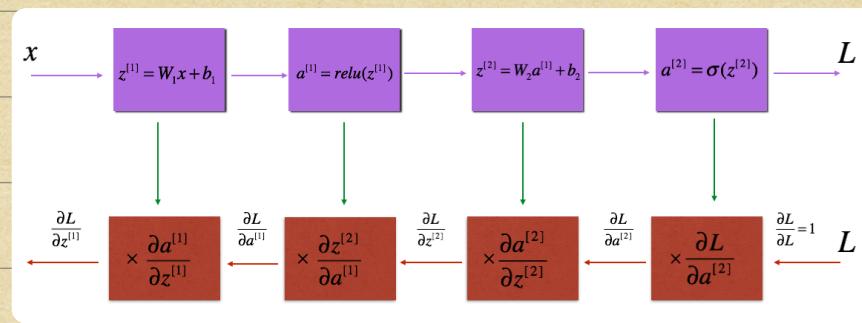


<1> 随机初始化参数，用 L-1 层线性 → ReLU 网络 + 一层输出层线性 → Sigmoid 网络组成

前向传播计算损失函数，后向传播计算各级导数。



<2> 上图为神经网络图



$$da^{[1]} = -\frac{y}{a} + \frac{(1-y)}{a(1-a)}$$

最末层初始化

<3> 上图为线性—ReLU, 线性—Sigmoid 框图

