



温州大學
WENZHOU UNIVERSITY

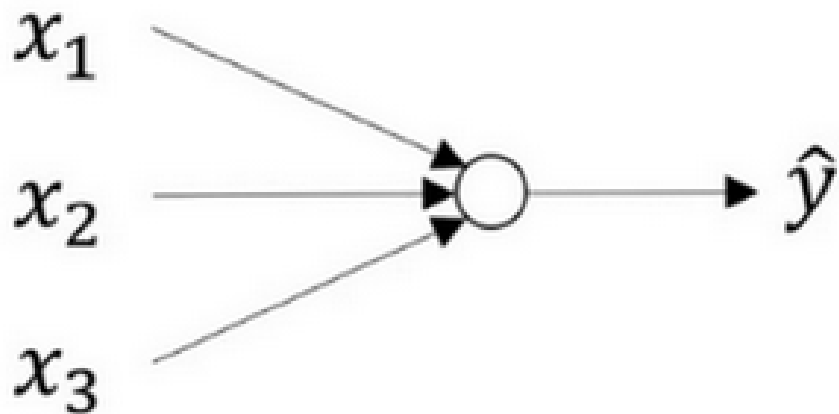
深度学习-深层神经网络

黄海广 副教授

2023年03月

神经网络的概念

2



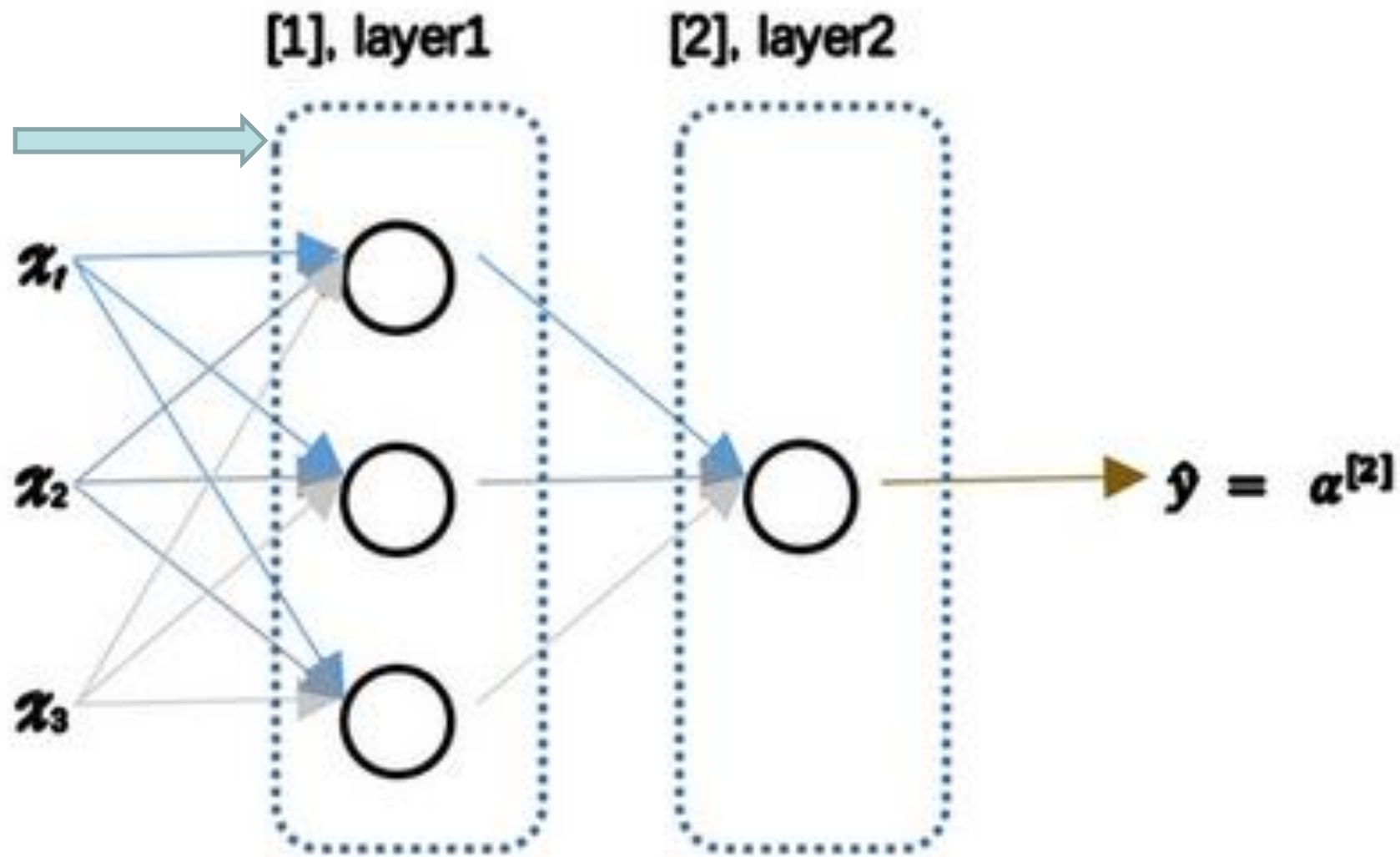
$$\left. \begin{matrix} x \\ w \\ b \end{matrix} \right\} \Longrightarrow z = w^T x + b \Longrightarrow \alpha = \sigma(z)$$

$$\Longrightarrow L(a, y)$$

神经网络的概念

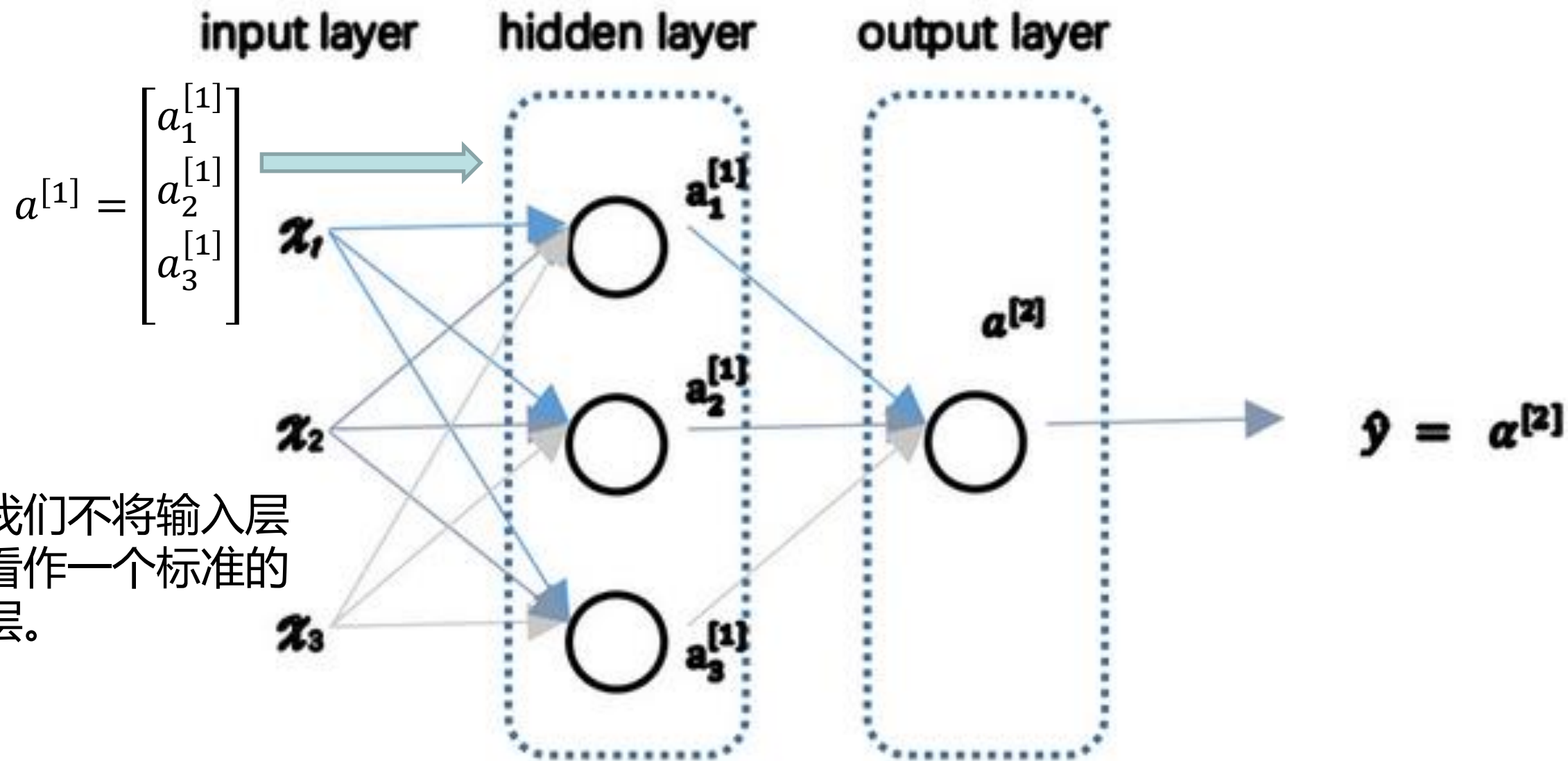
3

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix}$$



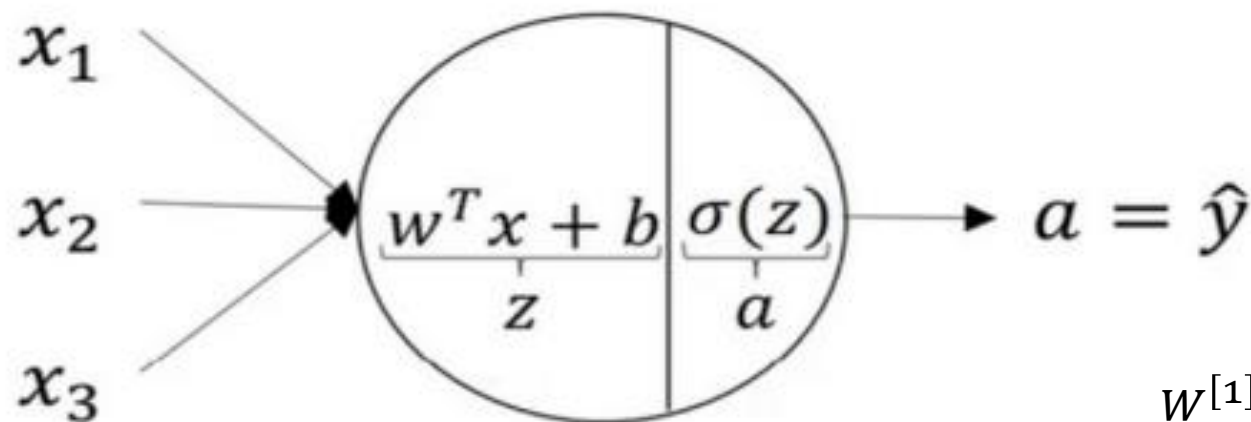
神经网络的概念

4



2.神经网络的向量化

5



$$z = w^T x + b$$

$$a = \sigma(z)$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \overbrace{\begin{bmatrix} \dots & W_1^{[1]T} & \dots \\ \dots & W_2^{[1]T} & \dots \\ \dots & W_3^{[1]T} & \dots \\ \dots & W_4^{[1]T} & \dots \end{bmatrix}}^{W^{[1]}} * \overbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}^{\text{input}} + \overbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}^{b^{[1]}}$$

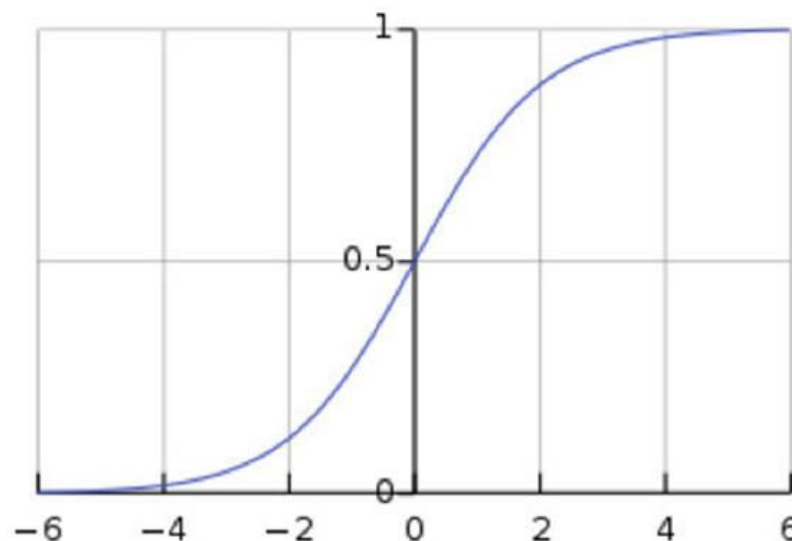
3. 激活函数

6

Sigmoid函数

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

$$\begin{aligned}\frac{d}{dz} g(z) &= \frac{1}{1+e^{-z}} \left(1 - \frac{1}{1+e^{-z}}\right) \\ &= g(z)(1 - g(z))\end{aligned}$$



sigmoid 函数

当 $\sigma(z)$ 大于等于0.5时, 预测 $y=1$

当 $\sigma(z)$ 小于0.5时, 预测 $y=0$

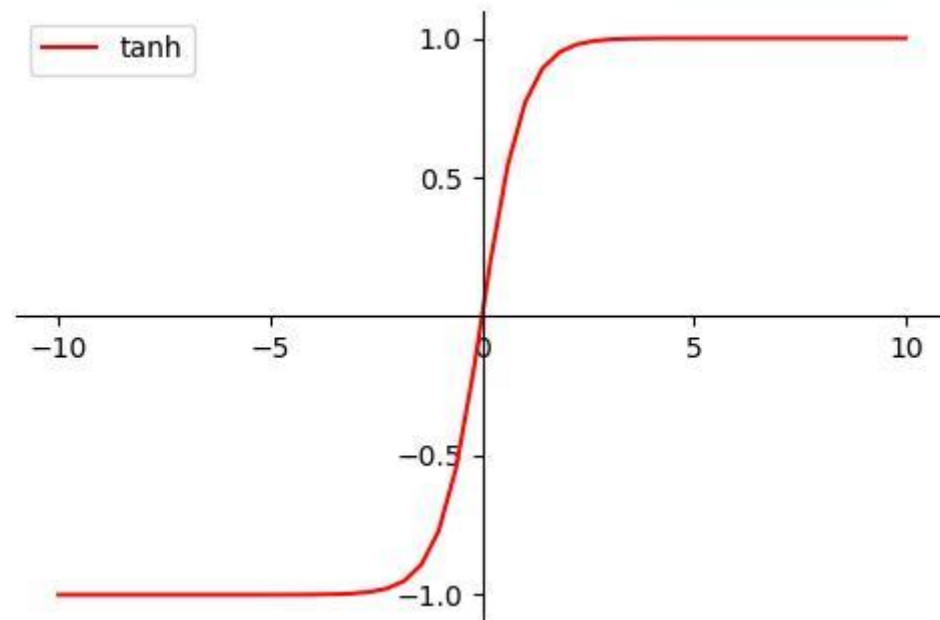
3. 激活函数

7

tanh函数

$$a = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\frac{d}{dz} g(z) = 1 - (\tanh(z))^2$$



tanh函数是**sigmoid**的向下平移和伸缩后的结果。对它进行了变形后，穿过了(0,0)点，并且值域介于+1和-1之间。

tanh函数是总体上都优于**sigmoid**函数的激活函数。

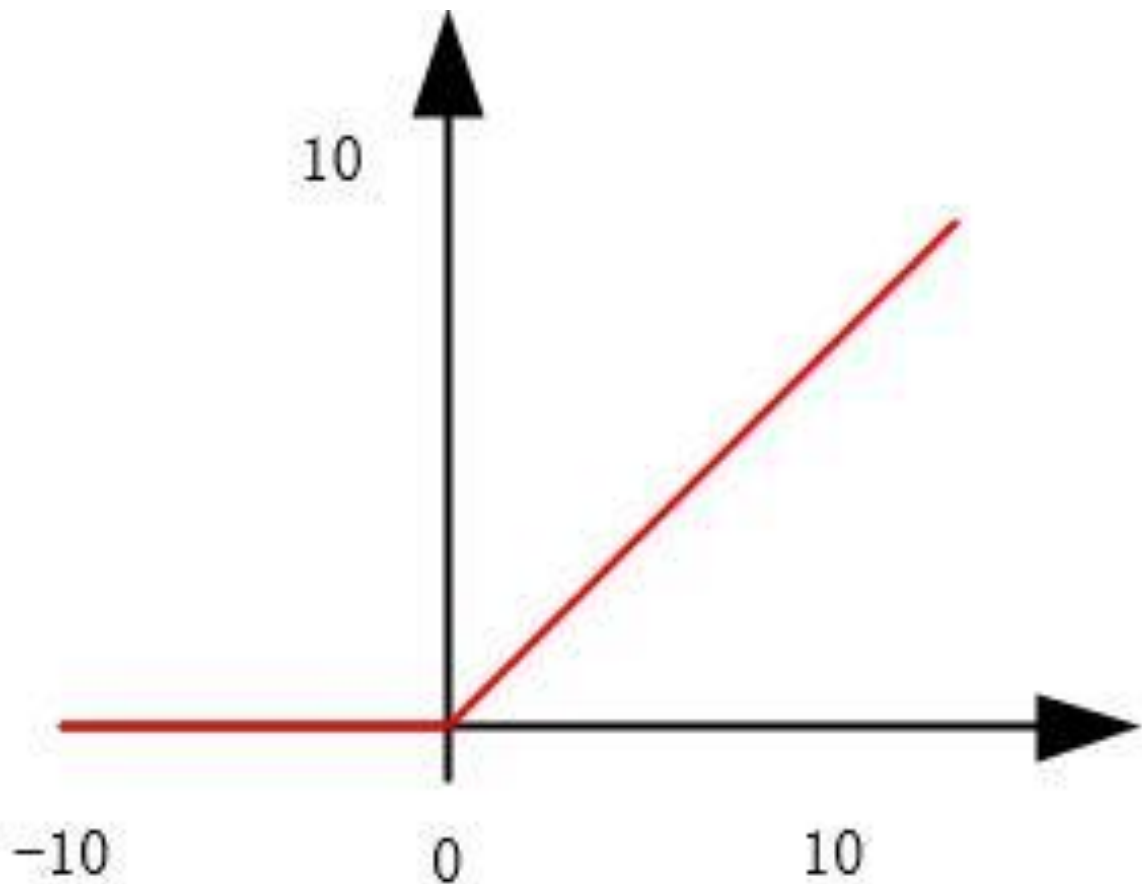
3. 激活函数

8

ReLU函数

$$a = \max(0, z)$$

在输入是负值的情况下，它会输出0，那么神经元就不会被激活。这意味着同一时间只有部分神经元会被激活，从而使得网络很稀疏，进而对计算来说是非常有效率的。

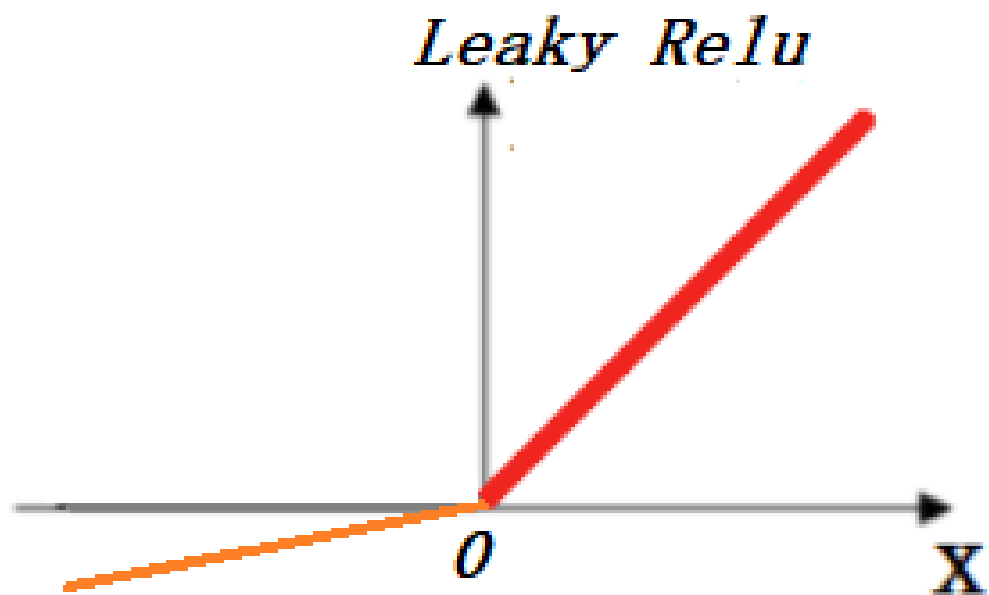


3. 激活函数

9

Leaky ReLU函数

$$a = \max(0.01z, z)$$



Leaky ReLU通常比Relu激活函数效果要好，
尽管在实际中Leaky ReLU使用的并不多。

3. 激活函数的使用场景

10

Sigmoid激活函数：除了输出层是一个二分类问题基本不会用它。

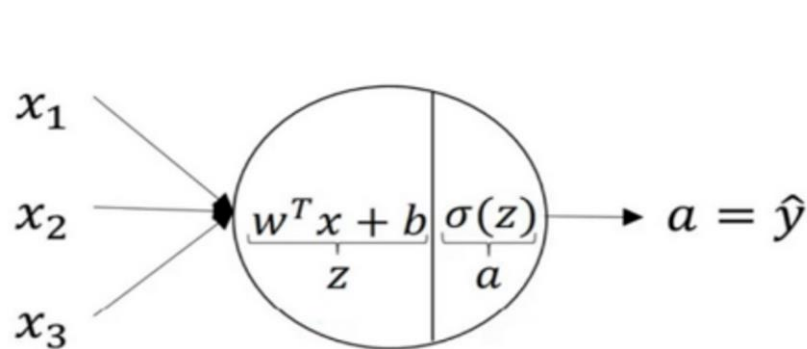
Tanh激活函数：**tanh**是非常优秀的，几乎适合所有场合。

ReLu激活函数：最常用的默认函数，，如果不确定用哪个激活函数，就使用**ReLu**或者**Leaky ReLu**。

4.反向传播算法

11

$$\left. \begin{matrix} x \\ w \\ b \end{matrix} \right\} \Rightarrow z = w^T x + b \Rightarrow a = \sigma(z) \Rightarrow L(a, y)$$



$$\left. \begin{matrix} x \\ w \\ b \end{matrix} \right\} \Leftarrow \begin{matrix} dz = da \cdot \overset{\curvearrowright}{g'(z)} \\ dw = dz \cdot \overset{\curvearrowright}{x}, db = dz \end{matrix}$$

\Leftarrow

$$z = w^T x + b$$

$$dz = da \cdot \overset{\curvearrowright}{g'(z)}, g(z) = \sigma(z), \overset{\curvearrowright}{\frac{dL}{dz}} = \frac{dL}{da} \cdot \frac{da}{dz}, \frac{d}{dz} g(z) = g'(z)$$

\Leftarrow

$$a = \sigma(z) \Leftarrow L(a, y)$$

$$da = \overset{\curvearrowright}{\frac{d}{da}} L(a, y) = (-y \log a - (1-y) \log(1-a))' = -\frac{y}{a} + \frac{1-y}{1-a}$$

4.反向传播算法

12

$$\text{因为} \frac{dL(a,y)}{dz} = \frac{dL}{dz} = \left(\frac{dL}{da}\right) \cdot \left(\frac{da}{dz}\right)$$

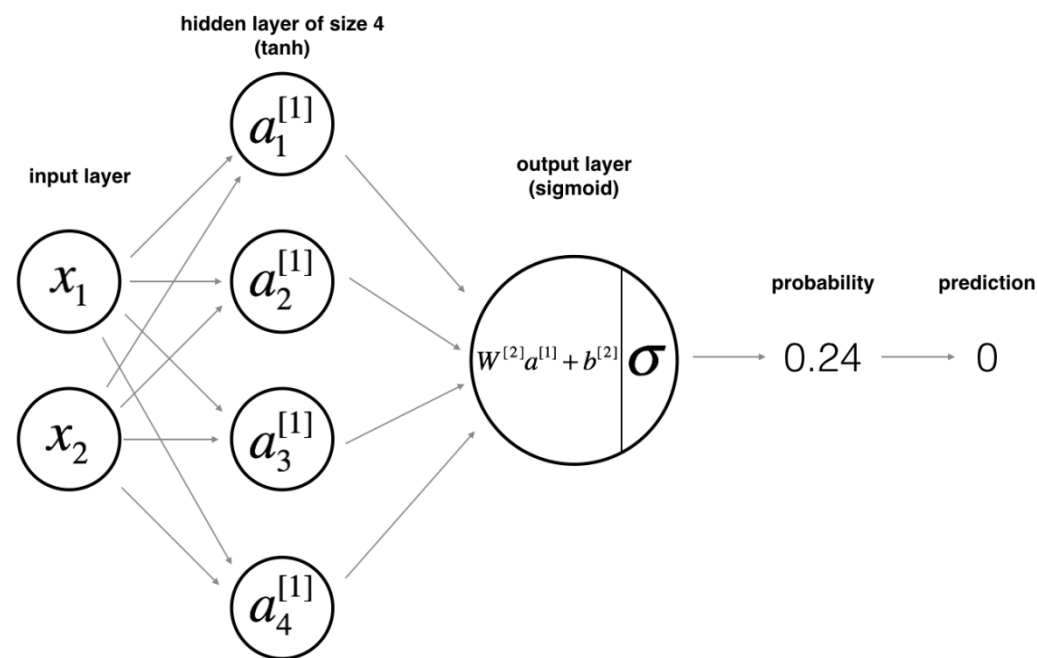
$$\text{并且} \frac{da}{dz} = a \cdot (1 - a),$$

$$\text{而} \frac{dL(a,y)}{da} = \frac{dL}{da} = \left(-\frac{y}{a} + \frac{(1-y)}{(1-a)}\right)$$

因此将这两项相乘，得到：

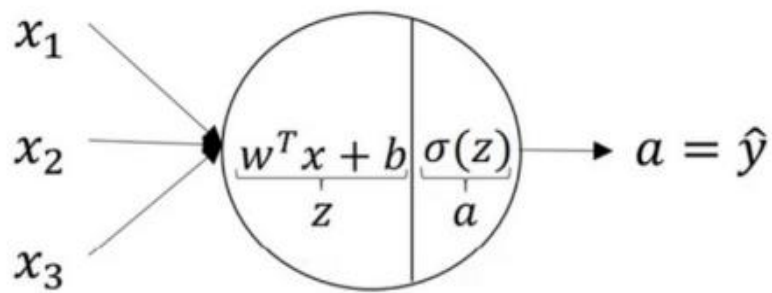
$$dz = \frac{dL(a,y)}{dz} = \frac{dL}{dz} = \left(\frac{dL}{da}\right) \cdot \left(\frac{da}{dz}\right)$$

$$= \left(-\frac{y}{a} + \frac{(1-y)}{(1-a)}\right) \cdot a(1 - a) = a - y$$



4.反向传播算法

13



$$z = w^T x + b$$

$$a = \sigma(z)$$

第一步，计算 $z_1^{[1]}$, $z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$ 。

第二步，通过激活函数计算 $a_1^{[1]}$, $a_1^{[1]} = \sigma(z_1^{[1]})$ 。

$a_2^{[1]}$ 、 $a_3^{[1]}$ 、 $a_4^{[1]}$ 的计算方式跟 $a_1^{[1]}$ 相似

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, a_1^{[1]} = \sigma(z_1^{[1]})$$

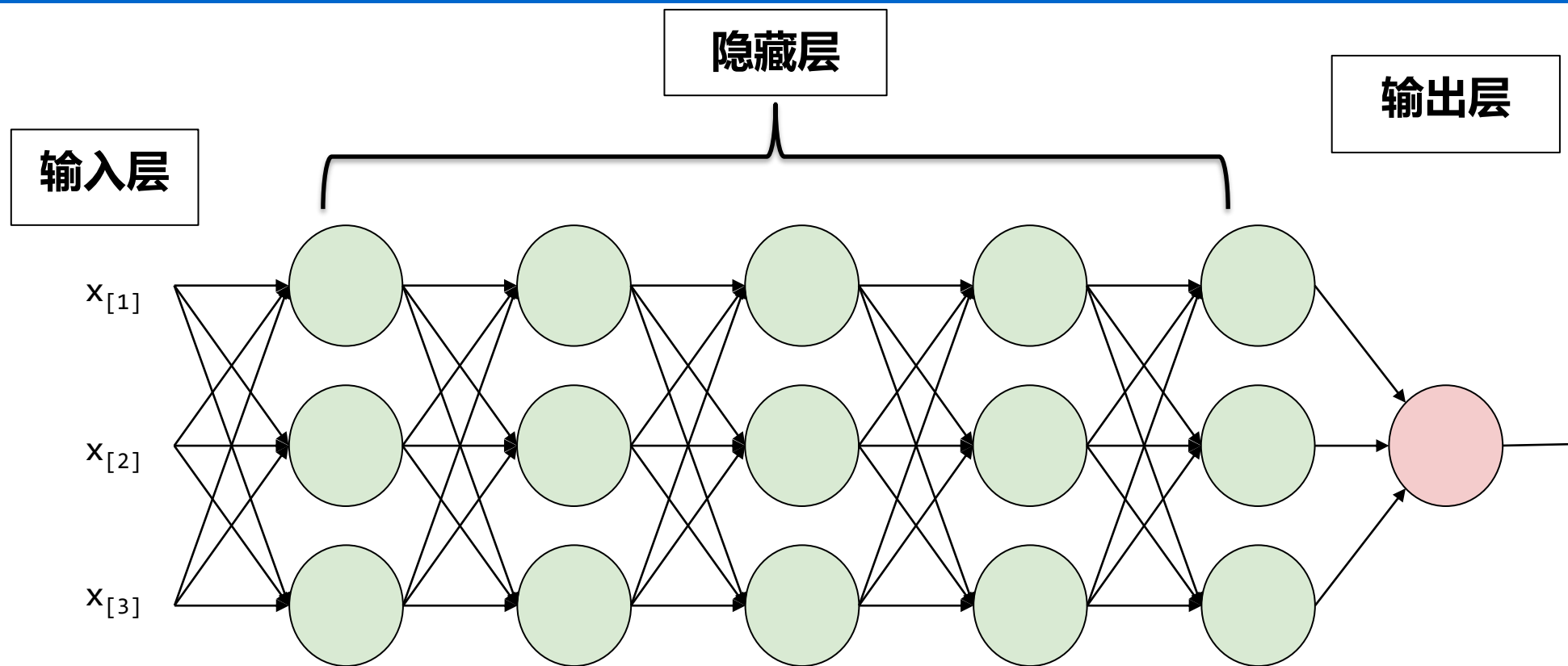
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, a_4^{[1]} = \sigma(z_4^{[1]})$$

4.反向传播算法

14



前向传播：

计算 $z^{[1]}$, $a^{[1]}$, 再计算 $z^{[2]}$, $a^{[2]}$, 最后得到**loss function**。

4.反向传播算法

15

反向传播:

向后推算出 $da^{[2]}$ ，然后推算出 $dz^{[2]}$ ，接着推算出 $da^{[1]}$ ，然后推算出 $dz^{[1]}$ 。我们不需要对 x 求导，因为 x 是固定的，我们也不是想优化 x 。向后推算出 $da^{[2]}$ ，然后推算出 $dz^{[2]}$ 的步骤可以合为一步：

$dz^{[2]} = a^{[2]} - y$, $dW^{[2]} = dz^{[2]}a^{[1]T}$ (注意：逻辑回归中；为什么 $a^{[1]T}$ 多了个转置： dw 中的 W 是一个列向量，而 $W^{[2]}$ 是个行向量，故需要加个转置)；

4.反向传播算法

16

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g[1]'(z^{[1]}) \text{ 注意: 这里的矩阵:}$$

$$W^{[2]} \text{ 的维度是: } (n^{[2]}, n^{[1]}).$$

$z^{[2]}$, $dz^{[2]}$ 的维度都是: $(n^{[2]}, 1)$, 如果是二分类, 那维度就是 $(1, 1)$ 。

$z^{[1]}$, $dz^{[1]}$ 的维度都是: $(n^{[1]}, 1)$ 。

证明过程:

其中 $W^{[2]T} dz^{[2]}$ 维度为: $(n^{[1]}, n^{[2]})$ 、 $(n^{[2]}, 1)$ 相乘得到 $(n^{[1]}, 1)$, 和 $z^{[1]}$ 维度相同, $g[1]'(z^{[1]})$ 的维度为 $(n^{[1]}, 1)$, 这就变成了两个都是 $(n^{[1]}, 1)$ 向量逐元素乘积。

4.反向传播算法

17

实现反向传播有个技巧，就是要保证矩阵的维度相互匹配。最后得到 $dW^{[1]}$ 和 $db^{[1]}$ ： $dW^{[1]} = dz^{[1]}x^T, db^{[1]} = dz^{[1]}$

可以看出 $dW^{[1]}$ 和 $dW^{[2]}$ 非常相似，其中 x 扮演了 $a^{[0]}$ 的角色， x^T 等同于 $a^{[0]T}$ 。

由： $Z^{[1]} = W^{[1]}x + b^{[1]}$ ， $a^{[1]} = g^{[1]}(Z^{[1]})$ 得到： $Z^{[1]} = W^{[1]}x + b^{[1]}, A^{[1]} = g^{[1]}(Z^{[1]})$

$$Z^{[1]} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ z^{[1]}(1) & z^{[1]}(2) & \vdots & z^{[1]}(m) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

注意：大写的 $Z^{[1]}$ 表示 $z^{[1]}(1), z^{[1]}(2), z^{[1]}(3) \dots z^{[1]}(m)$ 的列向量堆叠成的矩阵，以下类同。

4.反向传播算法

18

主要的推导过程: $dZ^{[2]} = A^{[2]} - Y$, $dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$

$$L = \frac{1}{m} \sum_i^m L(\hat{y}, y)$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1, \text{keepdims} = \text{True})$$

$$\underset{(n^{[1]}, m)}{dZ^{[1]}} = \underset{(n^{[1]}, m)}{W^{[2]T}} \underset{(n^{[1]}, m)}{dZ^{[2]}} * \underset{(n^{[1]}, m)}{g^{[1]'}(Z^{[1]})}$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} x^T$$

$$db^{[1]} = \frac{1}{m} \text{np.sum}(dZ^{[1]}, \text{axis} = 1, \text{keepdims} = \text{True})$$

4.反向传播算法

19

总结

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$

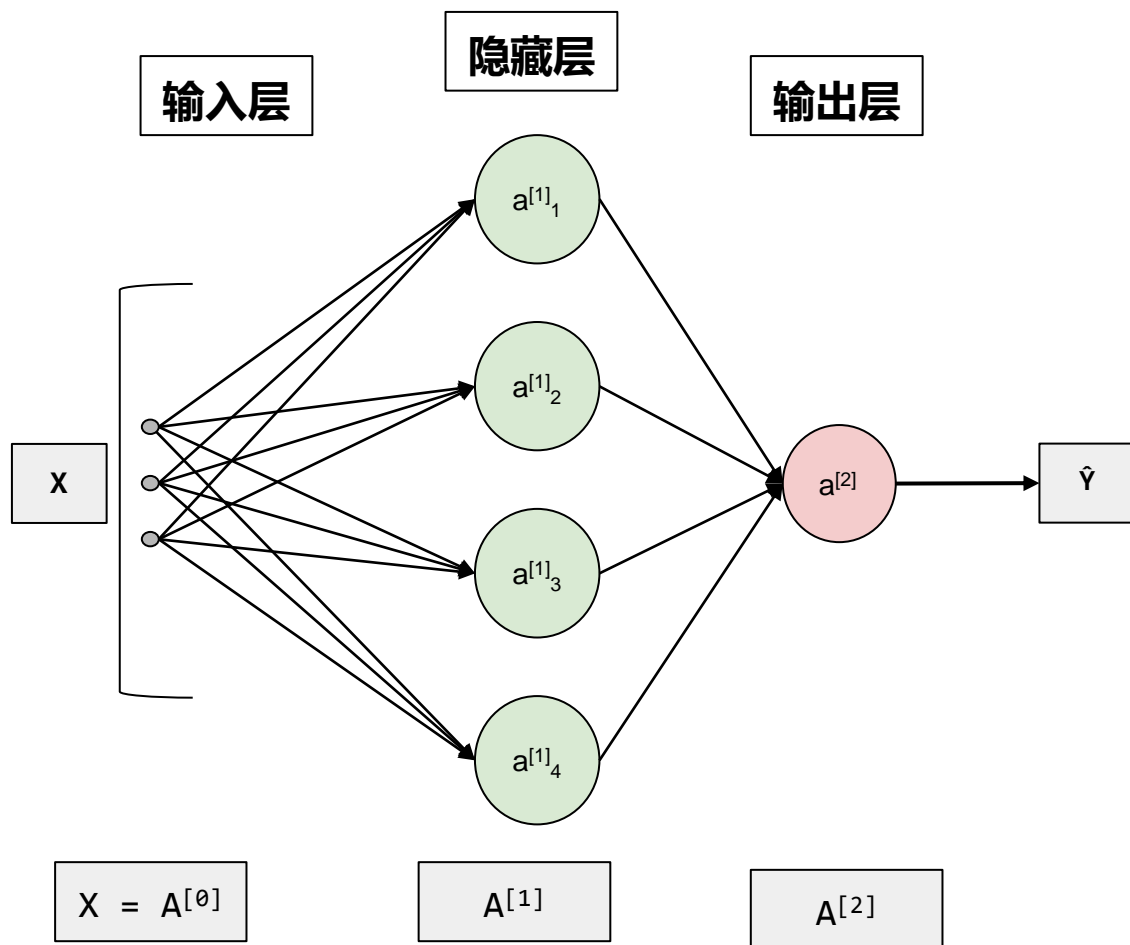
$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]'}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

5. 深层神经网络

20

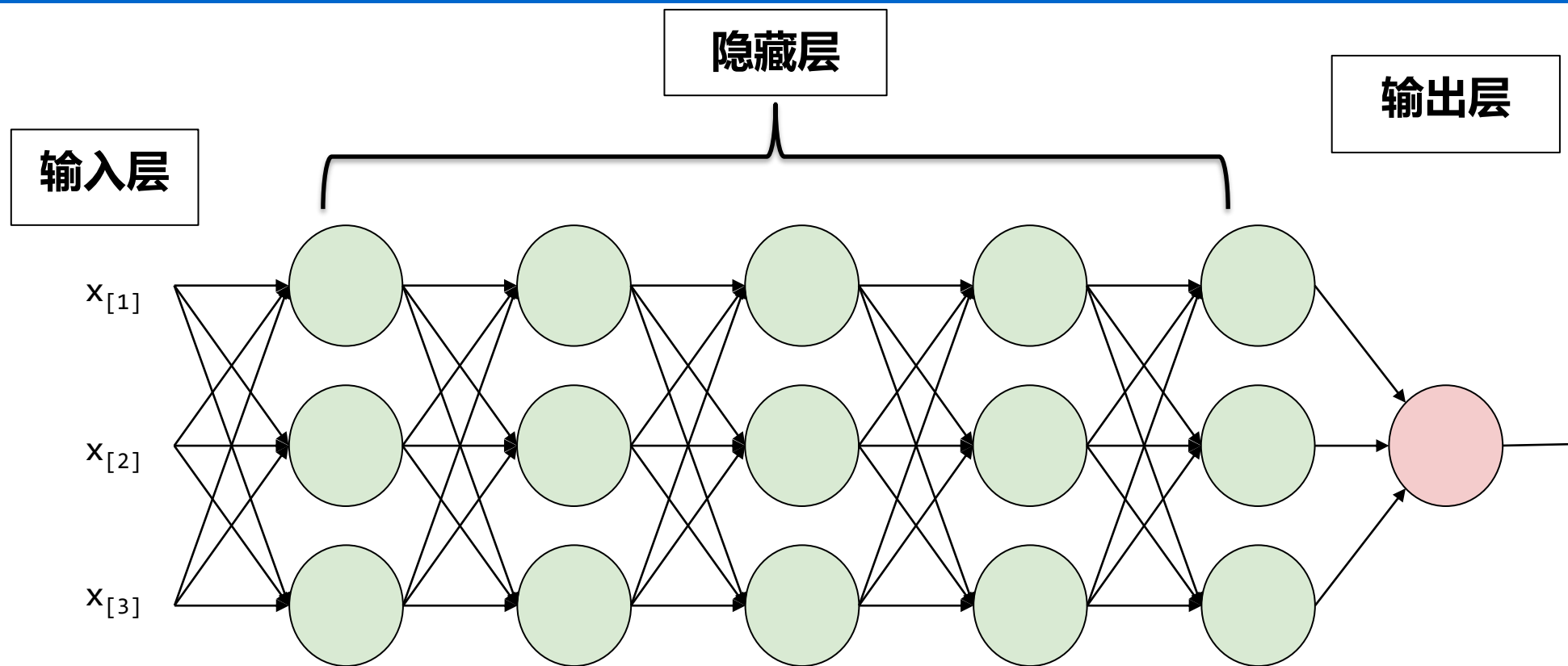


这是一个隐藏层的神经网络，当我们算神经网络的层数时，我们不算输入层，我们只算隐藏层和输出层。

左边这个网络是两层神经网络。

5. 深层神经网络

21



这是五个隐藏层的神经网络，隐藏层和输出层一共是六层，因此我们称这个是六层的神经网络。

前向传播

22

第一层需要计算 $z^{[1]} = w^{[1]}x + b^{[1]}$,

$a^{[1]} = g^{[1]}(z^{[1]})$ (x 可以看做 $a^{[0]}$)

第二层

$z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$, $a^{[2]} = g^{[2]}(z^{[2]})$

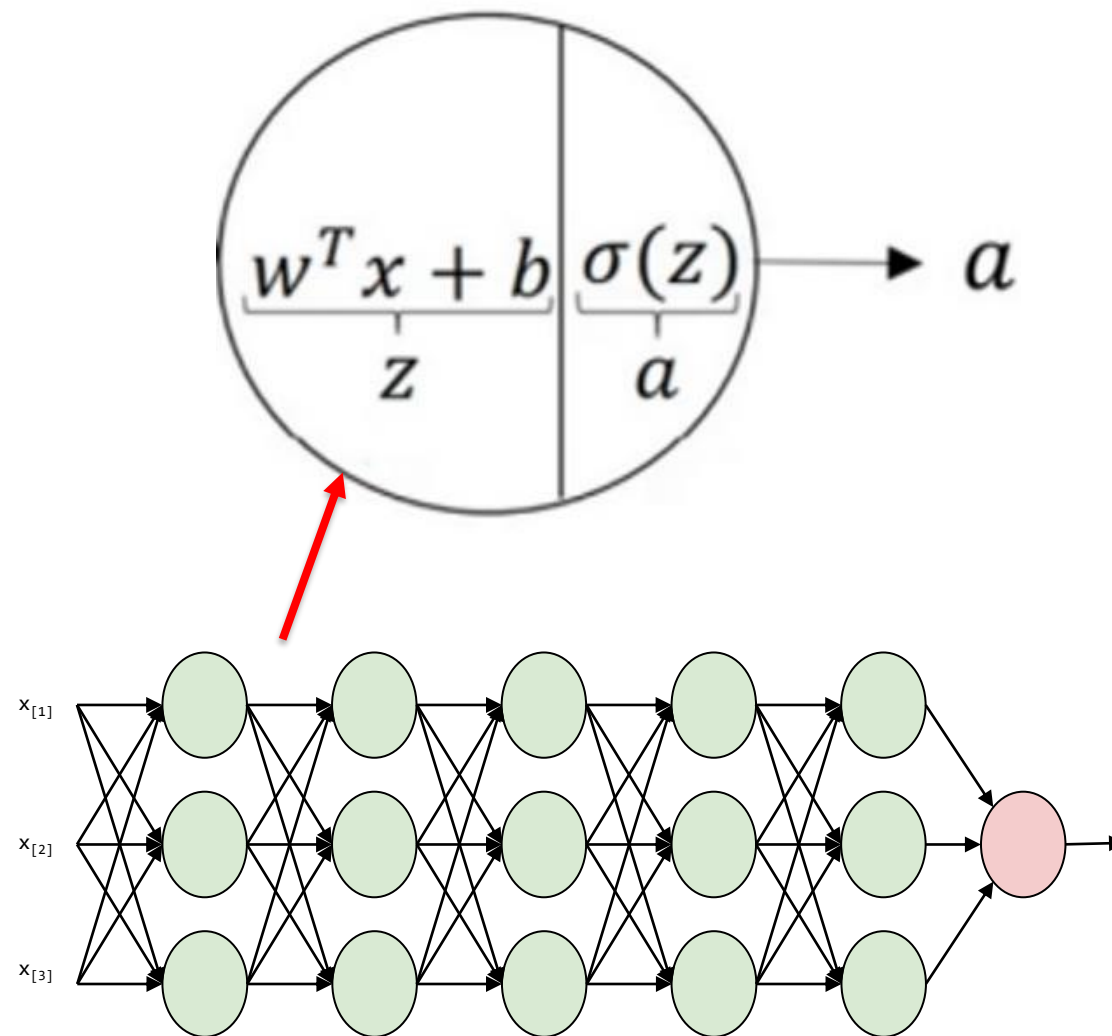
以此类推,

第四层为

$z^{[4]} = w^{[4]}a^{[3]} + b^{[4]}$, $a^{[4]} = g^{[4]}(z^{[4]})$

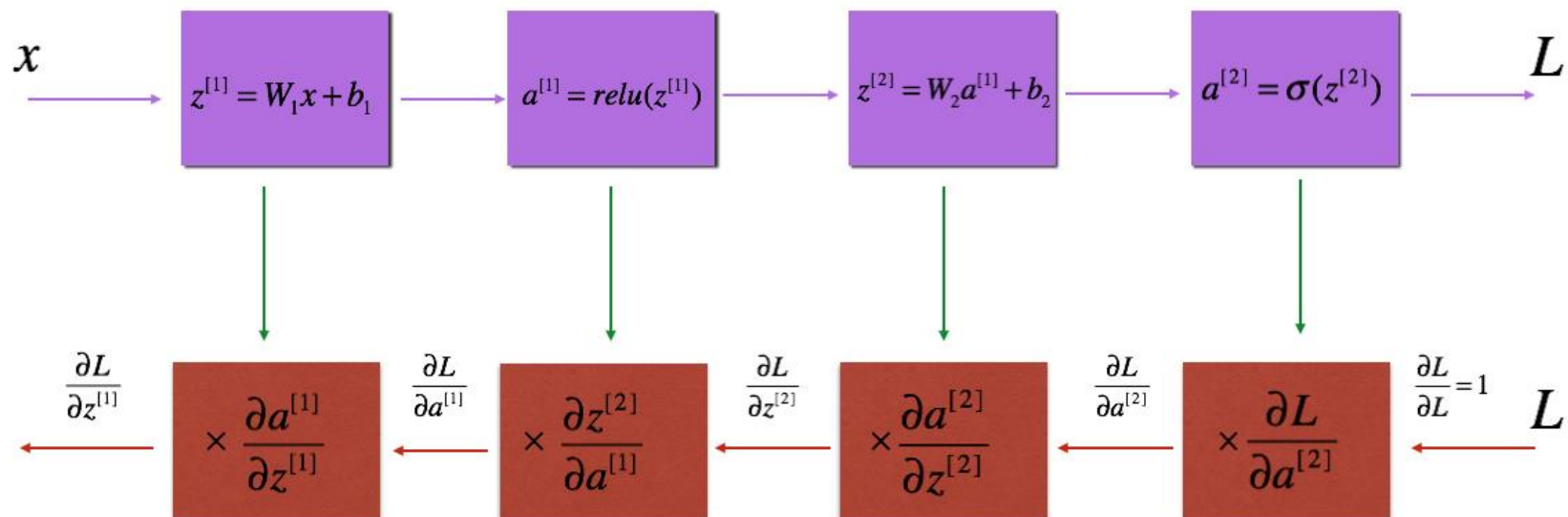
前向传播可以归纳为多次迭代

$z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]}$, $a^{[l]} = g^{[l]}(z^{[l]})$ 。



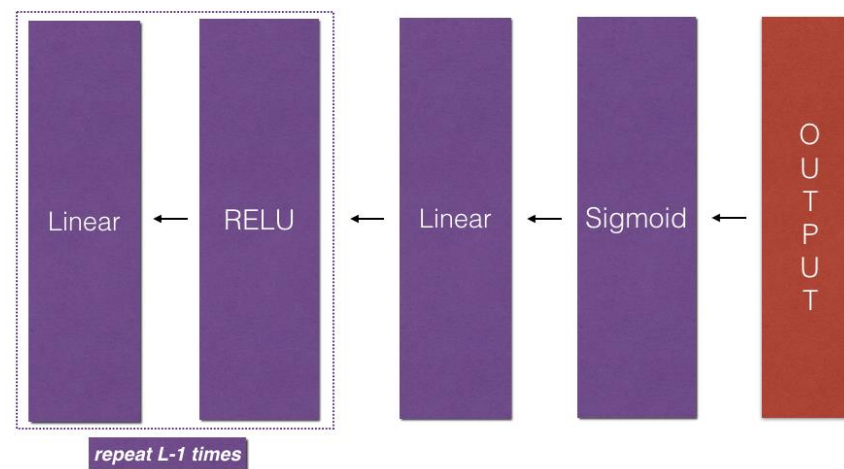
反向传播

23



前向传播和反向传播：
LINEAR->*RELU*->*LINEAR*->*SIGMOID*

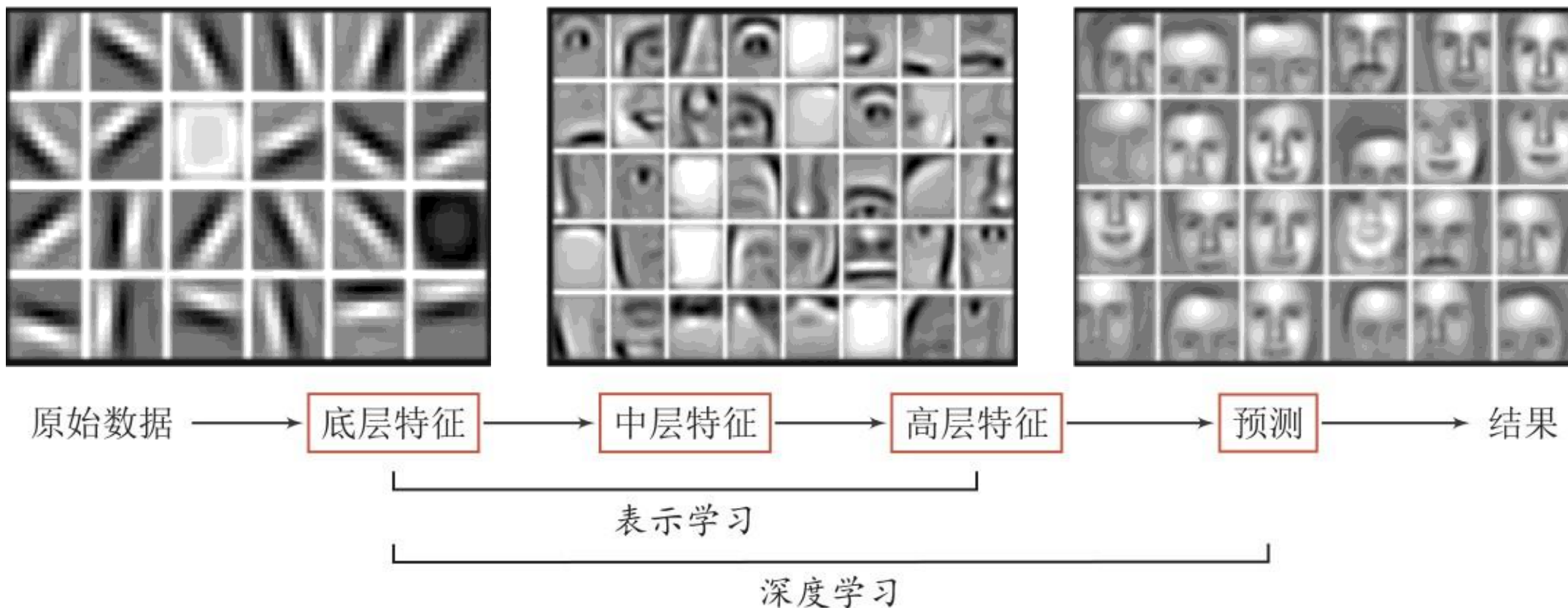
反向传播
重复L-1次



6.深度学习的表示

24

深度学习 = 表示学习 + 浅层学习



7.参数和超参数

25

什么是超参数?

比如算法中的**learning rate** a (学习率)、**iterations**(梯度下降法循环的数量)、 L (隐藏层数目)、 $n^{[l]}$ (隐藏层单元数目)、**choice of activation function** (激活函数的选择) 都需要你来设置, 这些数字实际上控制了最后的参数 W 和 b 的值, 所以它们被称作超参数。

7.参数和超参数

26

深度学习和大脑的关联性

深度学习和大脑有什么关联性吗？

关联不大。

1. IAN GOODFELLOW等, 《深度学习》, 人民邮电出版社, 2017
2. Andrew Ng, <http://www.deeplearning.ai>

谢谢!

