

# 深度学习

构造神经网络解决XOR问题

# 思考：意识来源于神经元吗？



来源于量子层面！



来源于细胞层面！

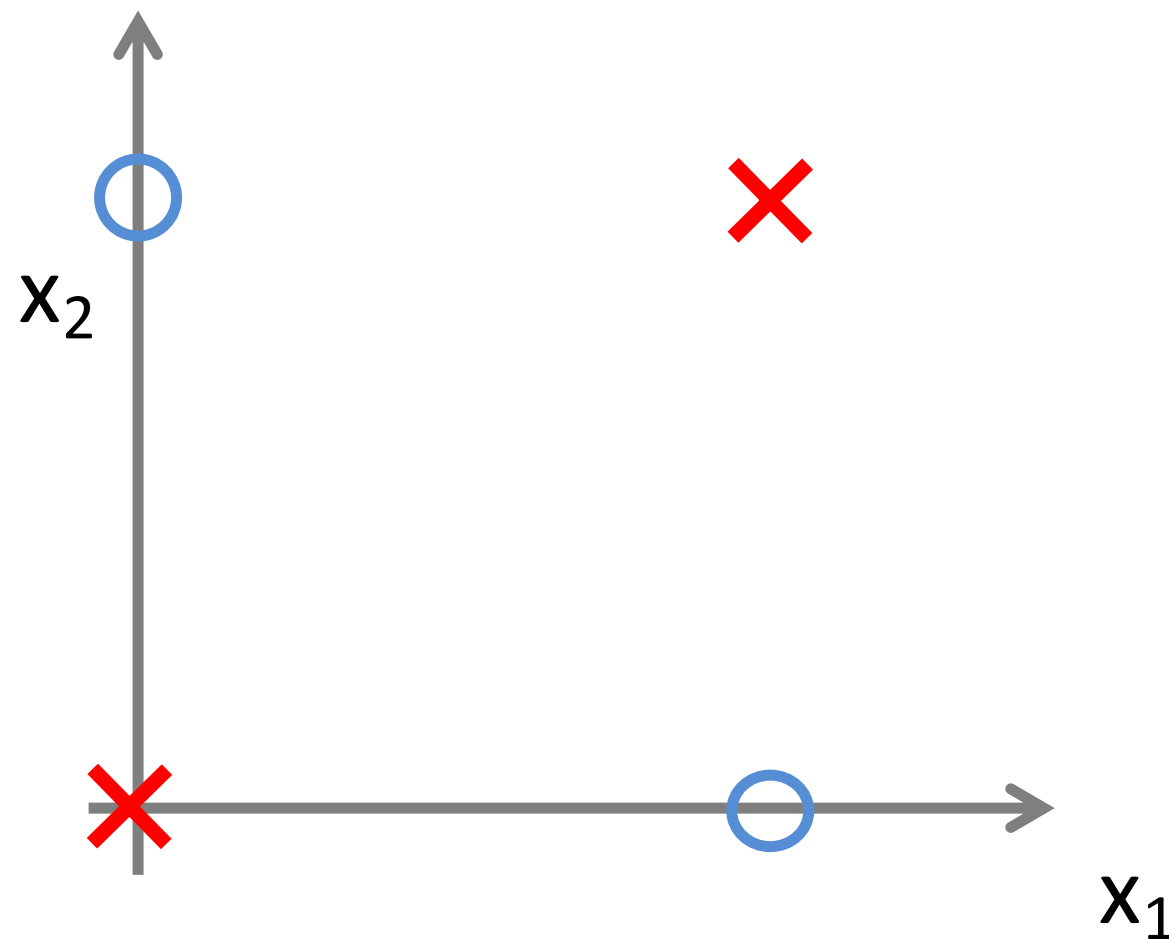
神经元构成神经网络，神经网络可以模拟任意函数。人或动物的任何活动都可以使用一个函数模拟，神经网络支配了人或动物的一切活动。即意识来源于神经网络。

# 直观理解

1. 任意过程均可以使用函数表达。
2. 任意连续函数均可以使用多项式逼近（泰勒级数）。
3. 任意多项式均可以使用加法来表示（差分机原理）。
4. 加法可以使用逻辑运算单元表示（XOR半加器）。

**结论：如果神经网络可以模拟半加器，就可以模拟任意过程。**

# XOR问题

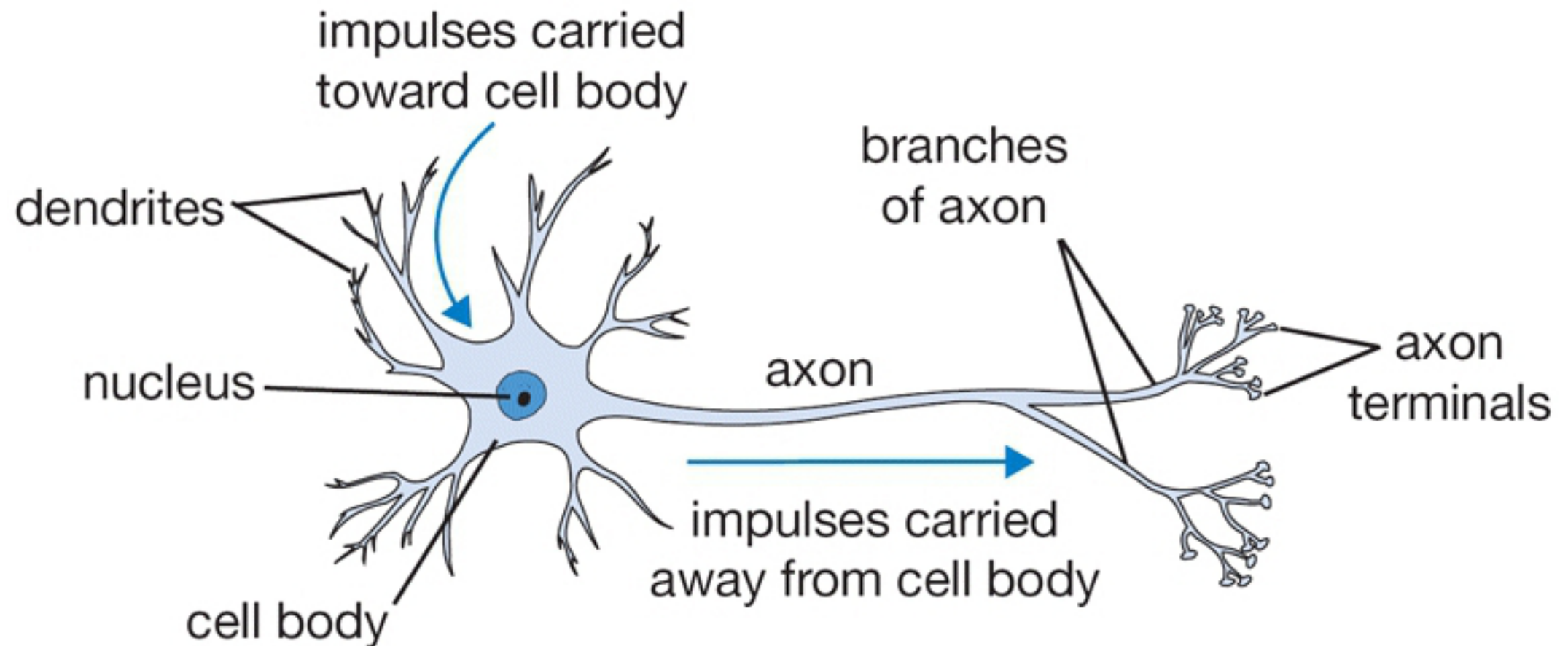


XOR即异或门，也是一个半加器。在图像上表示为线性不可分。

可以看到要想神经网络实现复杂的功能，首先必须让神经网络解决线性不可分问题。

从一个神经元开始，构造神经网络，尝试解决XOR问题。

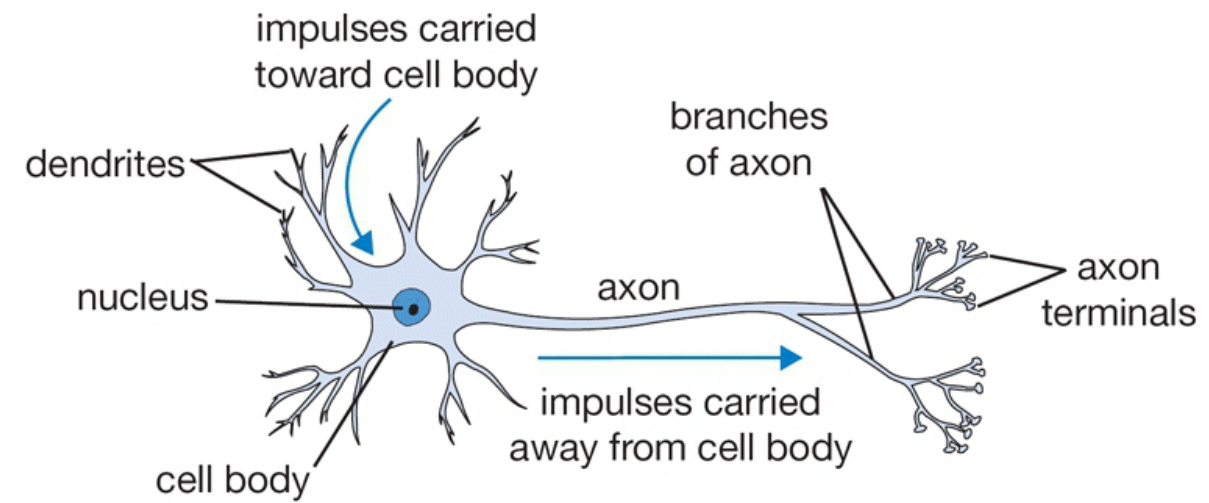
# 大脑中神经元的结构



# 神经元如何工作

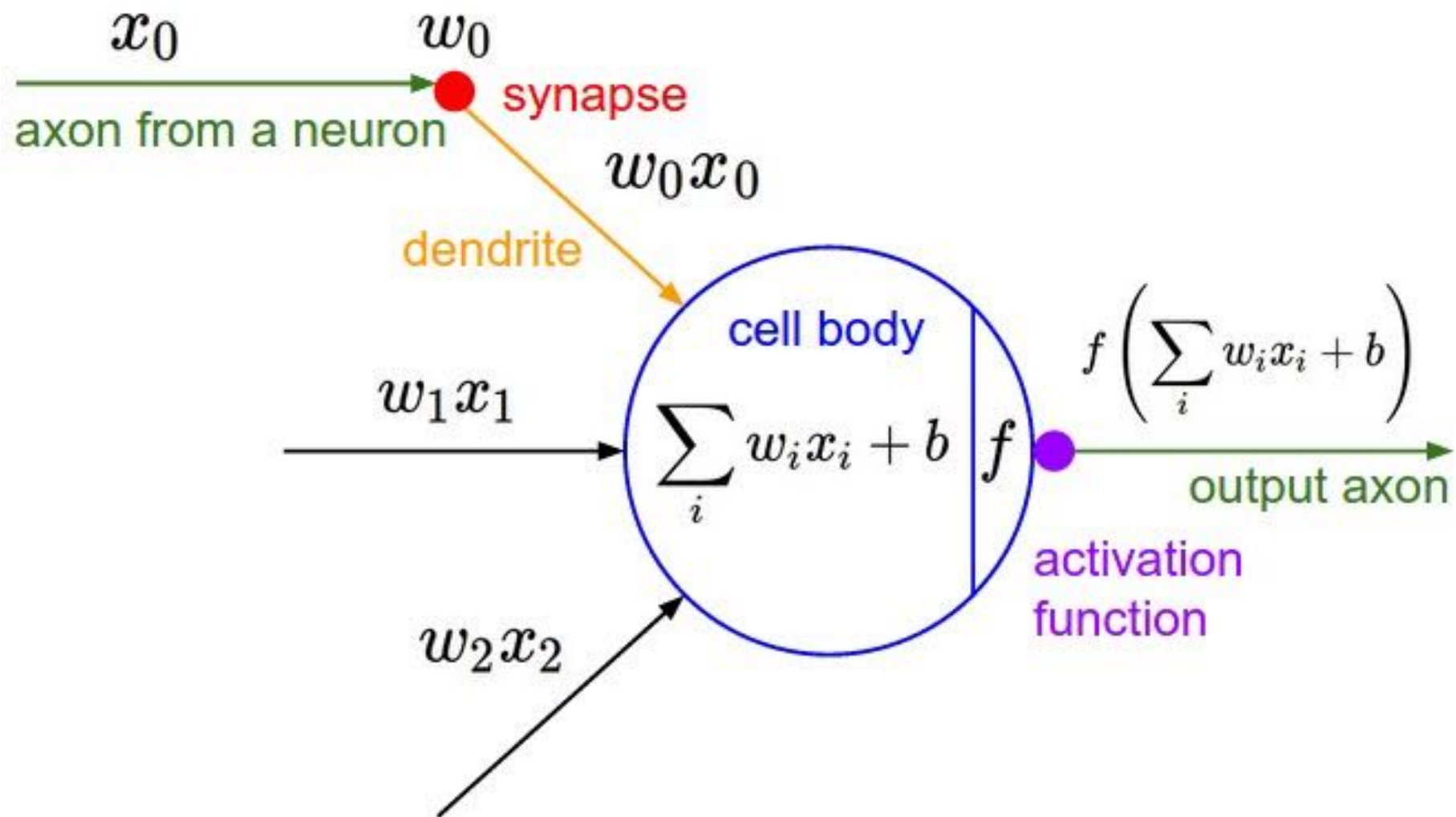
1. 生物神经元由树突传入信号(信号来源于上一个神经元细胞的树突或感受器)。
2. 进入胞体存留，达到阈值时，信号通过轴突传导出出去(给下一个神经元或肌肉或腺体)。神经元具有兴奋性和传导性。

# 神经元的特性



1. 树突存在多个，每个均可以传导电流进入胞体并存储。
2. 树突与上一个信号源的连接强弱程度不同，连接强度大则传导的电流大。
3. 胞体可以将每个树突传导来的电流存储在一起。
4. 每个神经元的胞体可以存储的电流多少不同，既触发传导下去的阈值不同。
5. 胞体传导出的电流强弱不一定与存储的电流相同。
6. 轴突传导出去的电流可以被多个树突检测到。

# 构造一个神经元

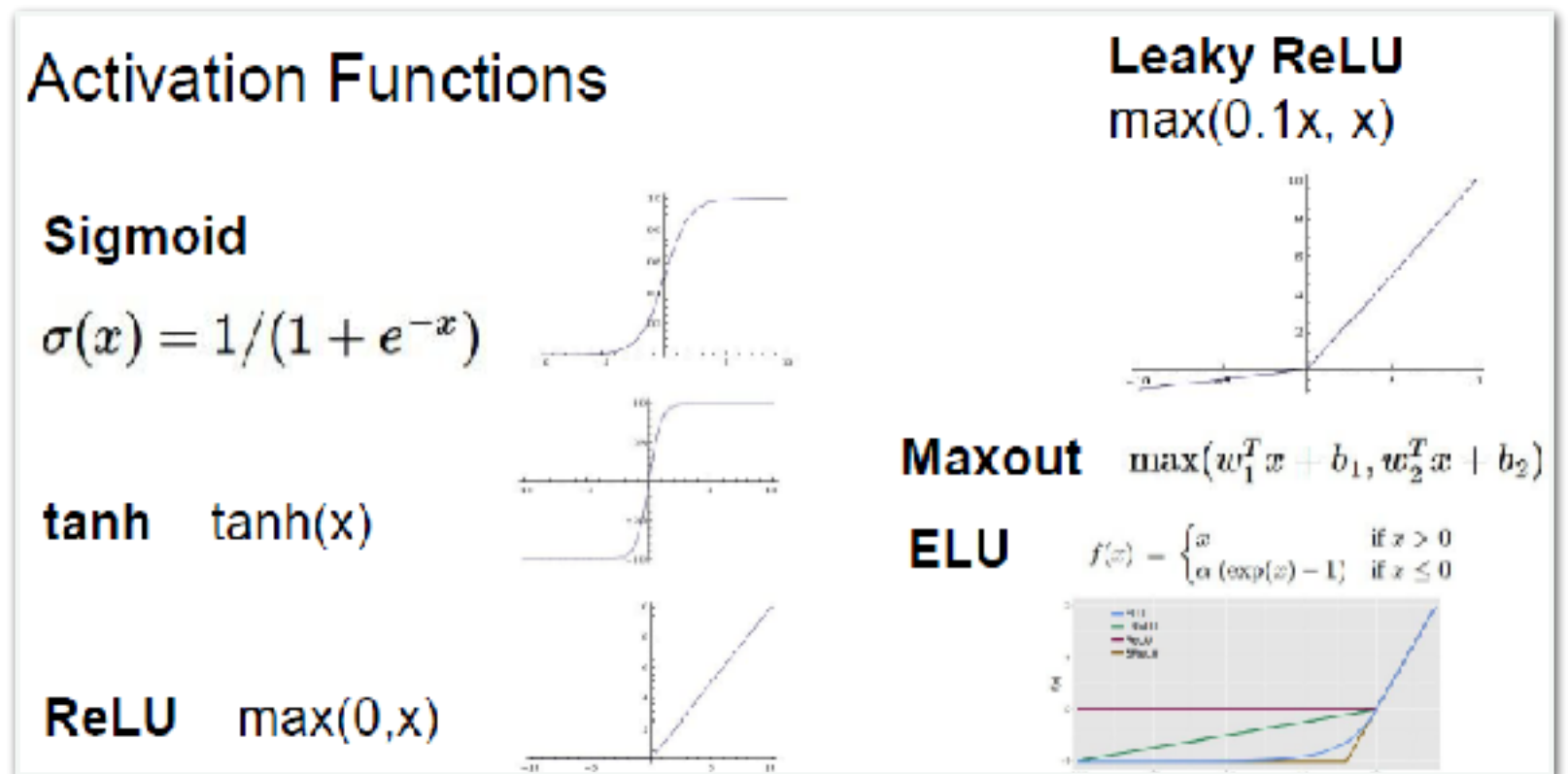


我们使用 $x$ 表示树突接收到的电流，使用 $w$ 表示树突的电阻，用sigma表示对输入的电流汇总，用 $f$ 表示对输入电流的转换（或者叫做激活函数），即轴突的输出。

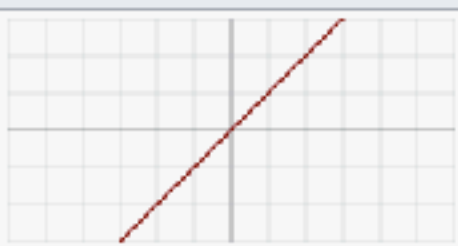

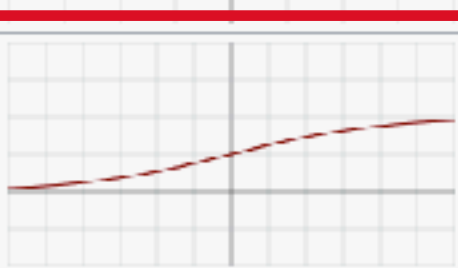
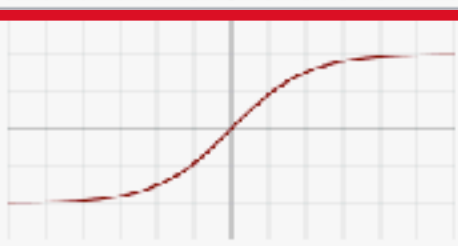
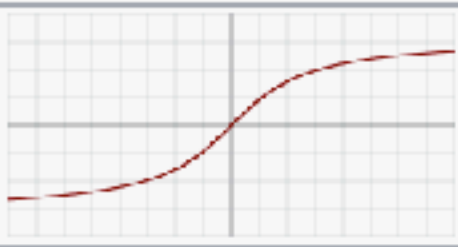
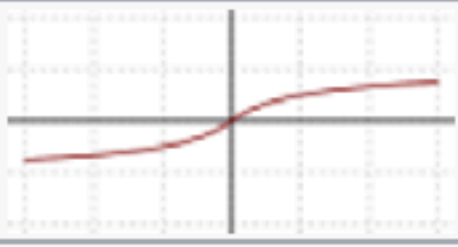
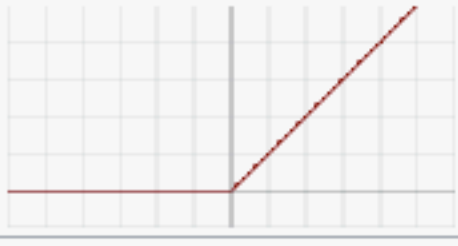
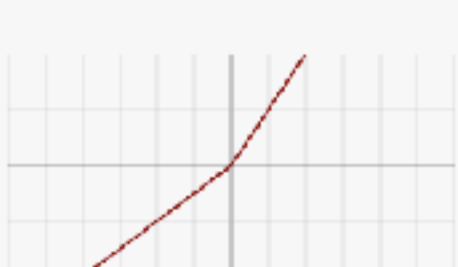


# 神经元的激活函数

- 阶梯函数
- 线性函数
- 饱和线性函数
- 对数S形函数
- 强制非负校正函数
- .....



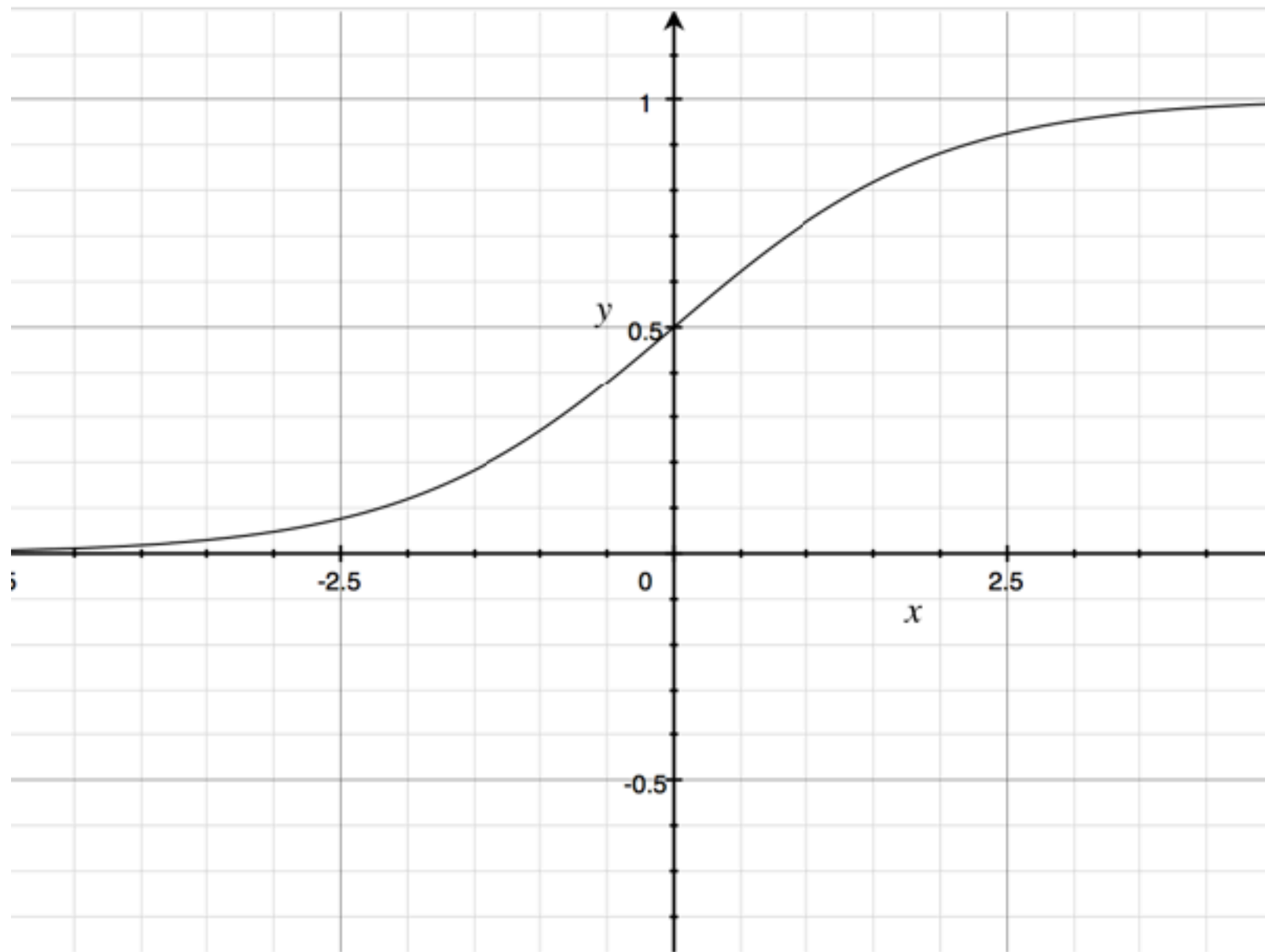
优秀的激活函数可以使神经网络更好的工作。设置线性激活函数或者不设置激活函数，会导致输出永远是输入的线性组合。为了能够解决上述提到的XOR问题，则必须选择非线性激活函数。

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign [7][8]		$f(x) = \frac{x}{1 +  x }$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU)[10]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

早期最常用激活函数

# Sigmoid激活函数

$$y = \frac{1}{1+e^{-x}}$$



优点：

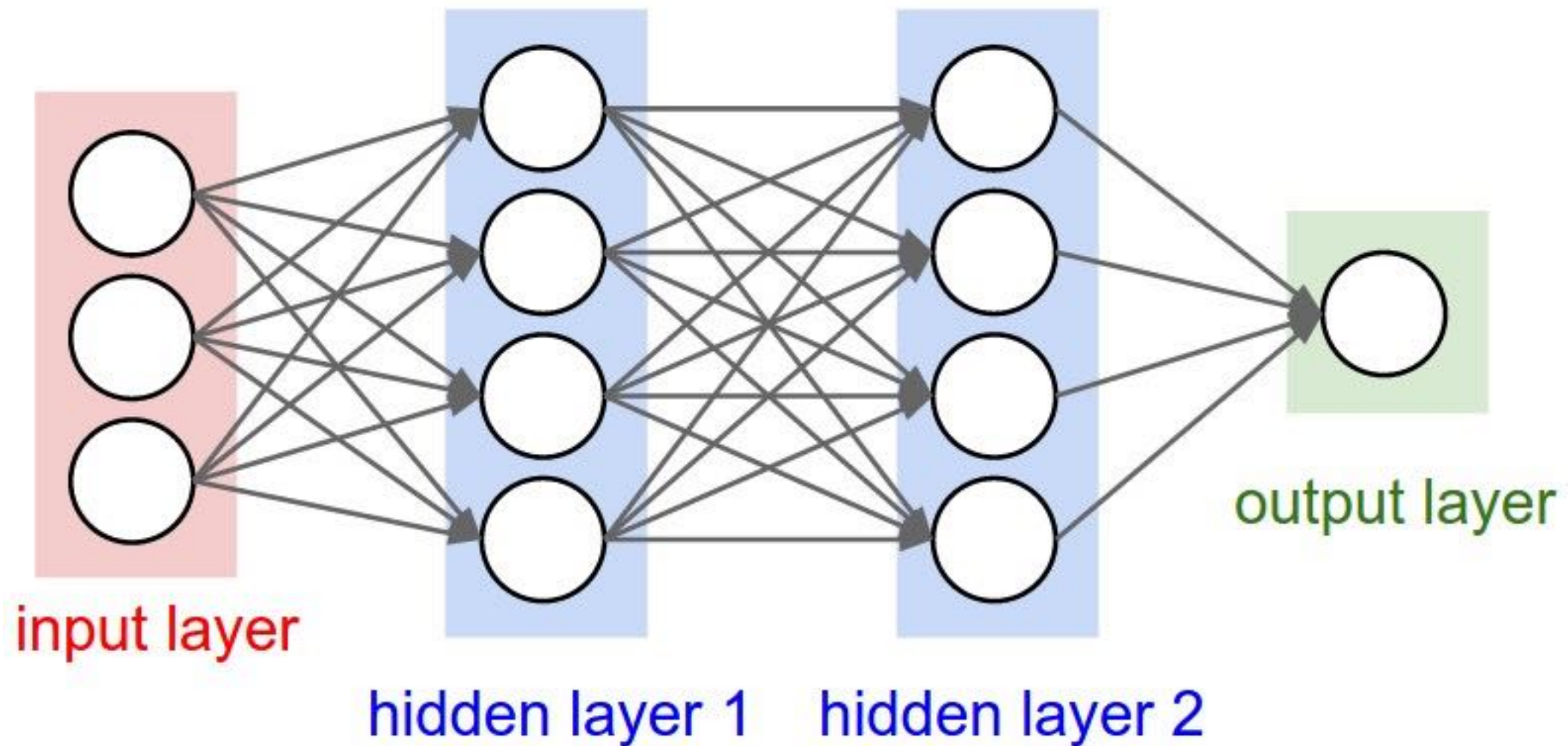
定义域为R，值域为(0,1)。  
可以用来做二分类。  
类似于生物神经元。

缺点：

计算量比较大。  
求导涉及除法。  
会导致梯度消失。

**sigmoid神经元可以模拟简单的逻辑回归。**

# 构造神经网络



图中每一个圆圈代表一个神经元(unit)。多个神经（一竖行）元组成一个层（layer）。层与层的神经元之间全连接。第一层我们称之为输入层，最后一层为输出层，其余为隐藏层。神经网络是一个有向无环图。

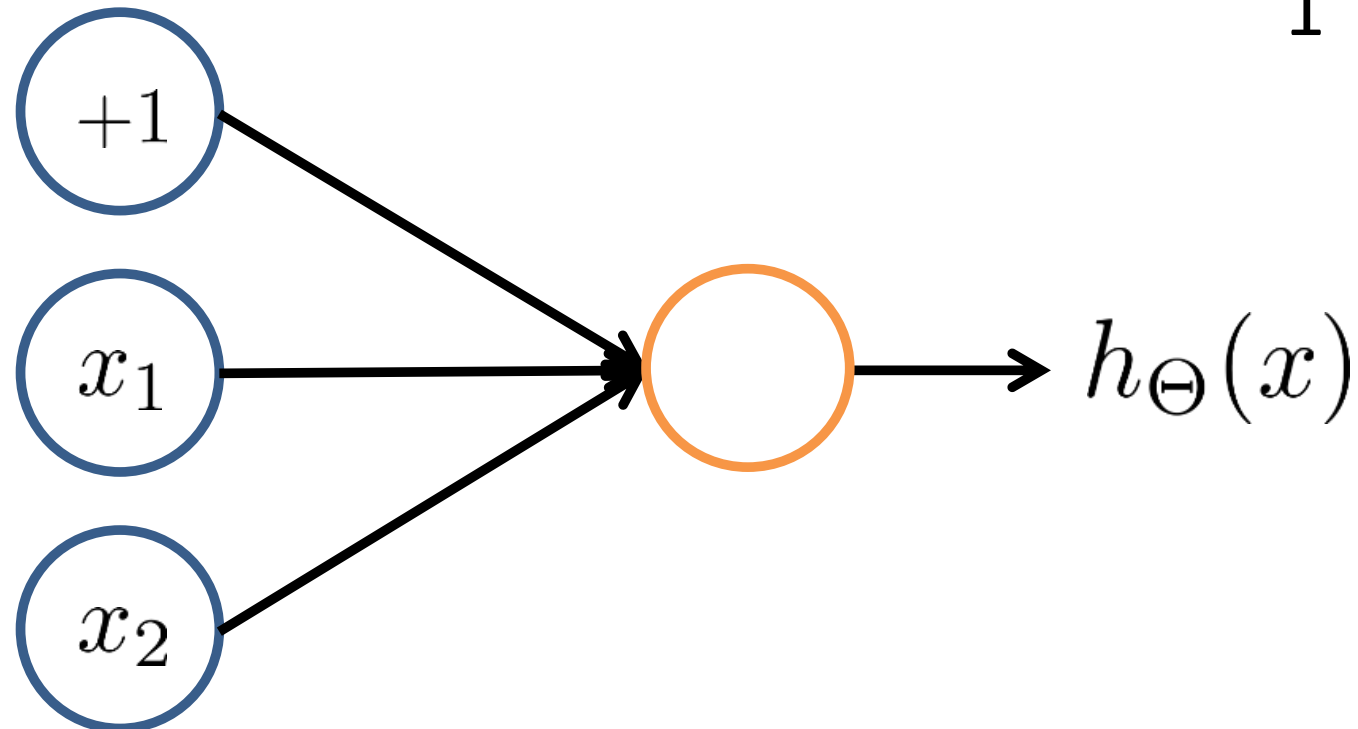
# 利用ANN构造逻辑运算单元

# 构造AND逻辑运算单元

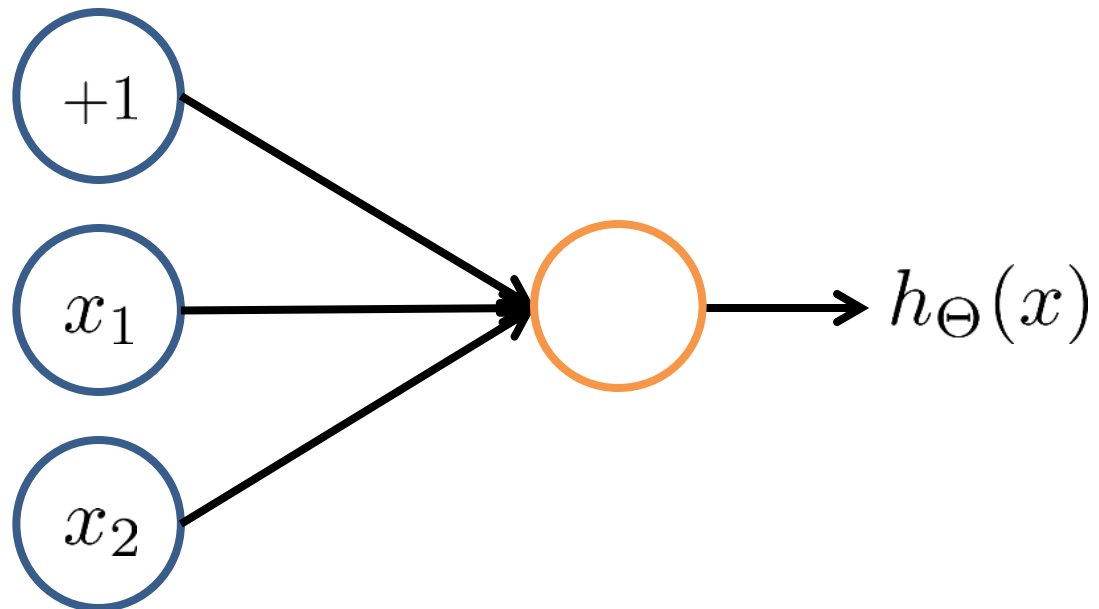
$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ AND } x_2$$

$x_1$	$x_2$	$h_{\Theta}(x)$
0	0	
0	1	
1	0	
1	1	

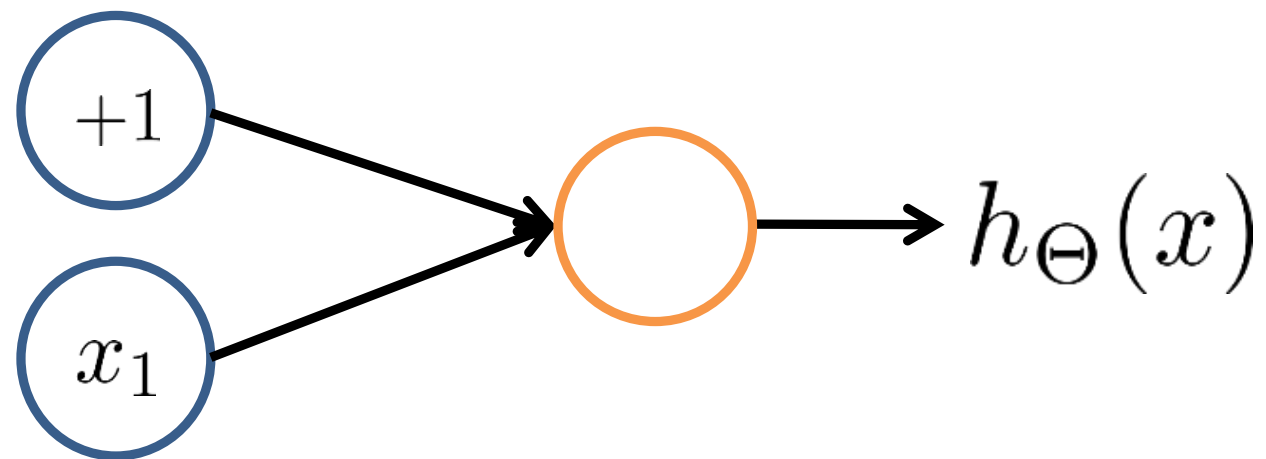


# 构造逻辑运算单元OR



$x_1$	$x_2$	$h_{\Theta}(x)$
0	0	
0	1	
1	0	
1	1	

# 构造逻辑运算单元NOT

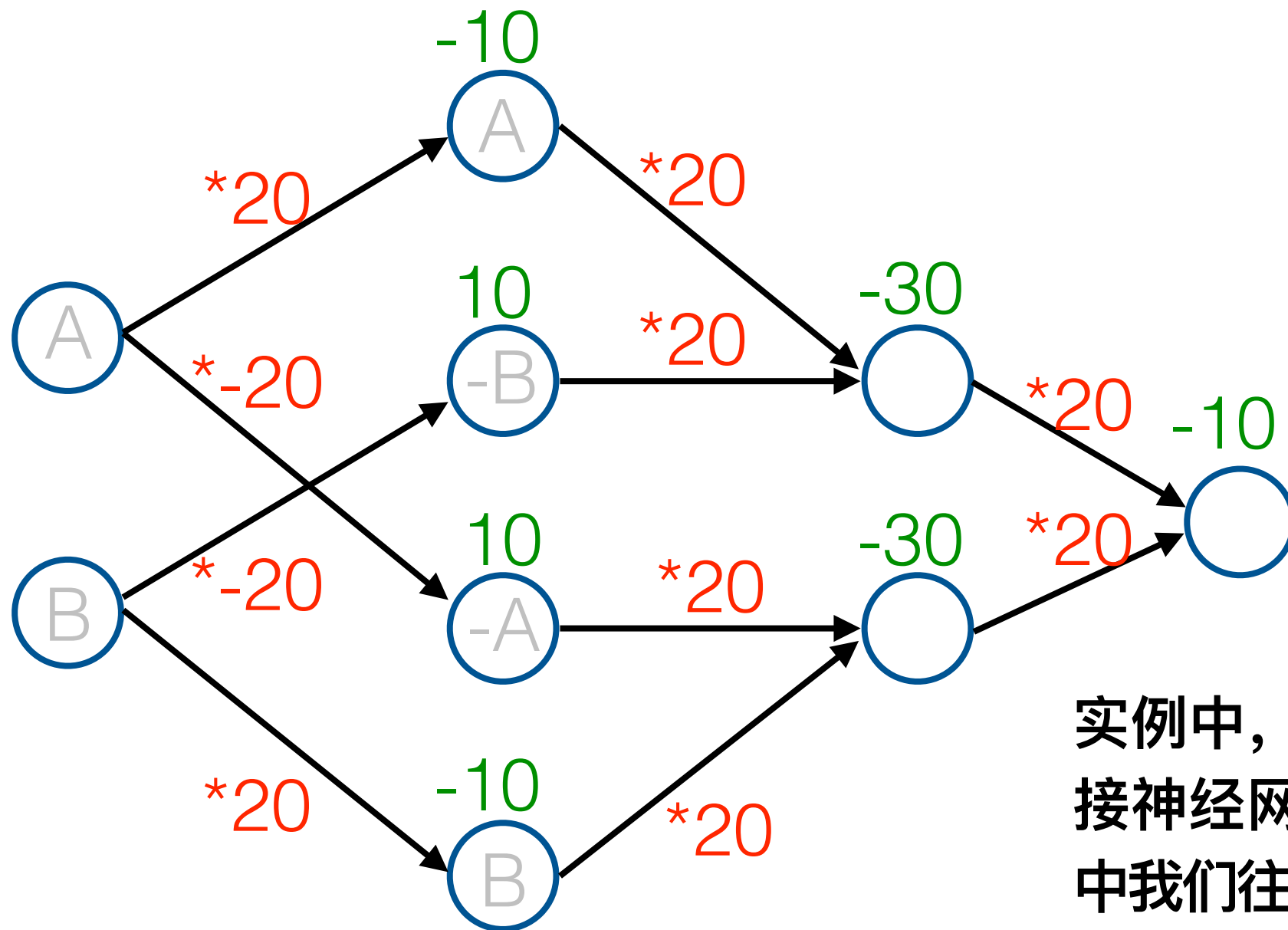


$x_1$	$h_{\Theta}(x)$
0	
1	



# XOR如何构造?

$$A \text{ xor } B = (A \text{ and } (\text{not } B)) \text{ or } ((\text{not } A) \text{ and } B)$$



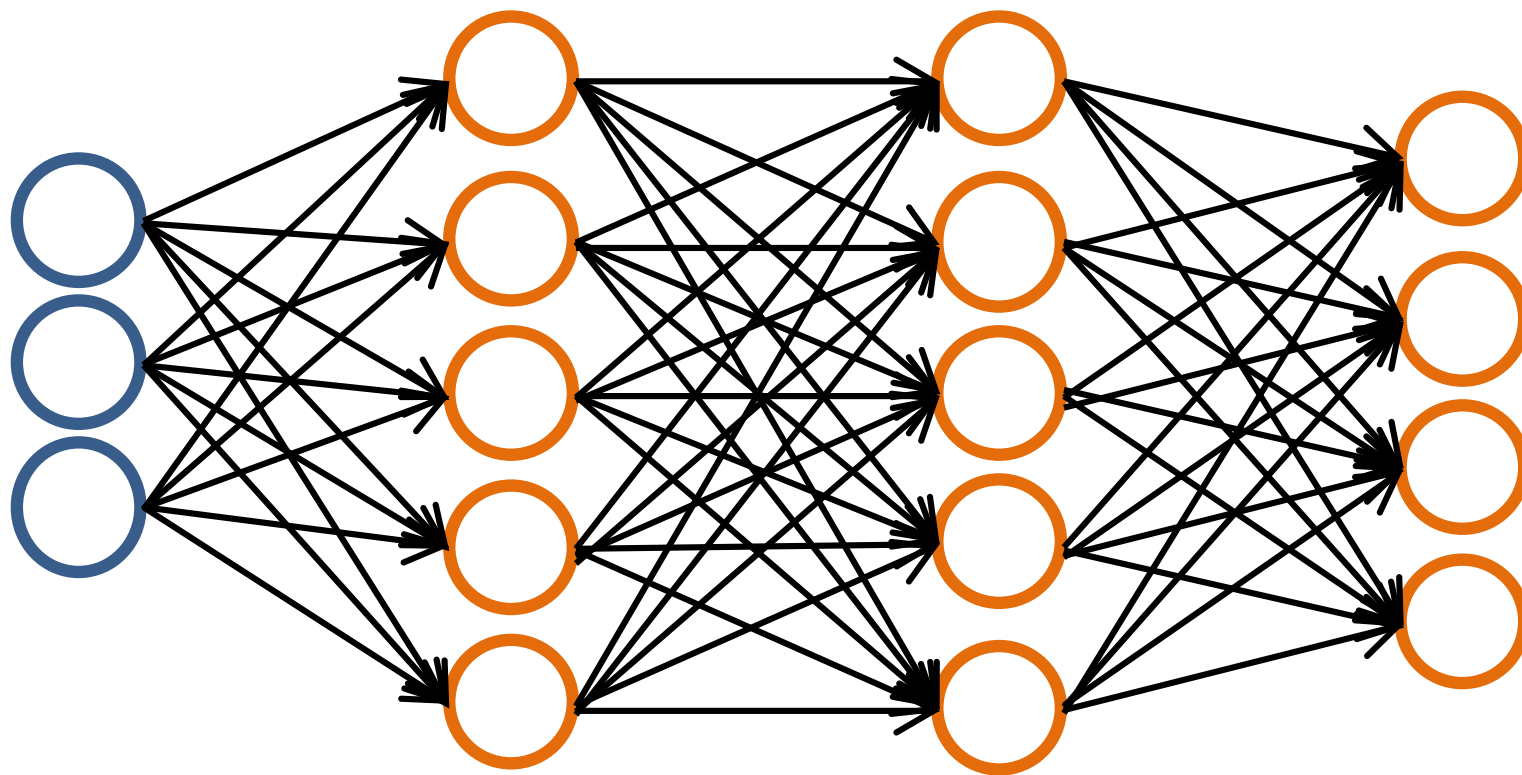
输入不同输出1

$x_1$	$x_2$	$h_{\Theta}(x)$
0	0	0
0	1	1
1	0	1
1	1	0

实例中，我们并非使用的是全连接神经网络，但是为了方便实际中我们往往使用全连接神经网络。

可以看到利用一些神经元可以轻松的构造出解决XOR问题的神经网络。同样的我们还可以构造出更加复杂的结构，来解决更加复杂的问题。而现在我们知道只要能够构造出可以叠加的半加器，就可以构造出能够以任意精度逼近任意函数的神经网络。

# 多分类问题神经网络



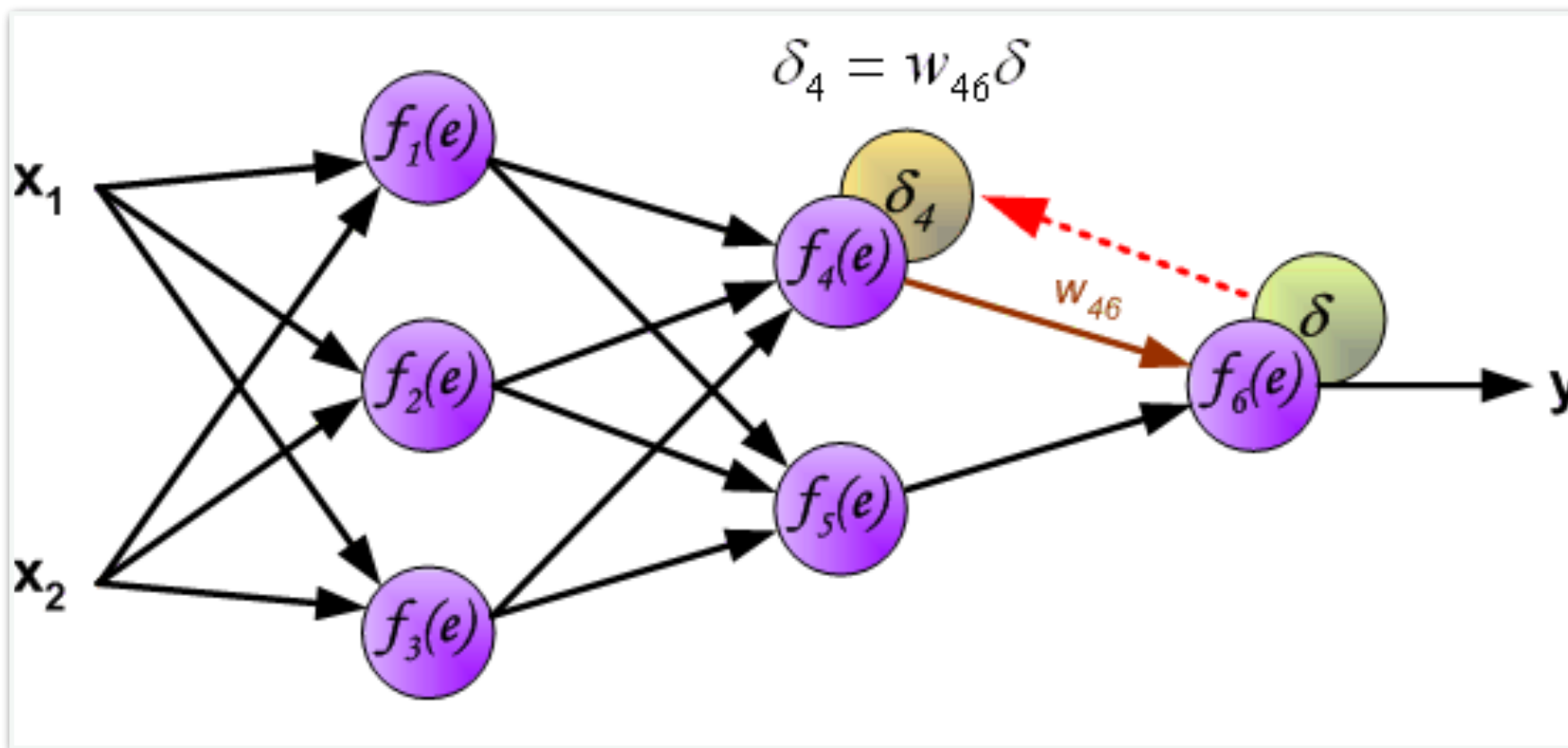
$$h_{\Theta}(x) \in \mathbb{R}^4$$

sigmoid神经元可以利用值域的特殊性来解决二分类问题。然而多分类问题，sigmoid神经元并不合适，这时候需要在输出层使用其它激活函数。

# 参数设置

手动设置参数?

NO!



实际中，我们并不手动设计神经网络的结构和参数。而是使用**反向传播算法**自动求参数。

# 小节

- 神经元的作用就是接收信号，变换信号，传出信号。
- 人工神经元接收多个输入，汇总，由激活函数处理后输出。
- 常用激活函数S型函数、双曲正切函数、修正线性单元。
- Sigmoid函数的值域决定了其可以做二分类。
- 神经网络是分层结构的。包括输入层、隐藏层、输出层
- 神经网络可以构造出逻辑单元并解决XOR问题。

# Python实现神经网络

# THANKS