# 生命科学中的机器学习

# 深度学习入门 (1)

黄　强

复旦大学 生命科学学院

# 三次课程的主要内容

1 深度学习的基本概况

2 感知机模型与单层人工神经网络

3 多层神经网络与误差反向传播算法

4 深度卷积神经网络与应用

# 本次课程内容

1 深度学习的基本概况

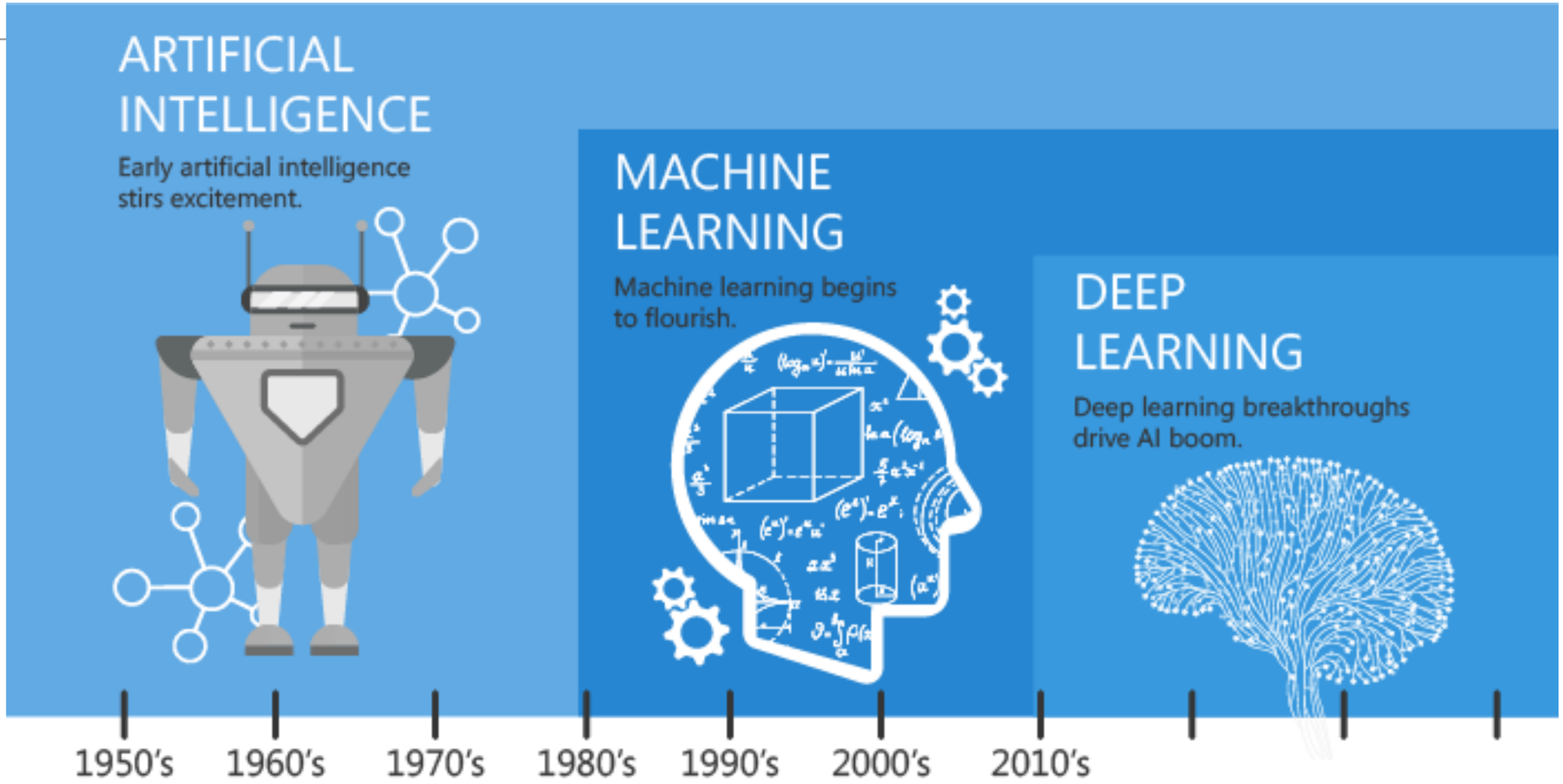
2 感知机模型与单层人工神经网络

3 多层神经网络与误差反向传播算法

4 深度卷积神经网络与应用

# 1.1 什么是深度学习？What is deep learning?

(from *Internet*)

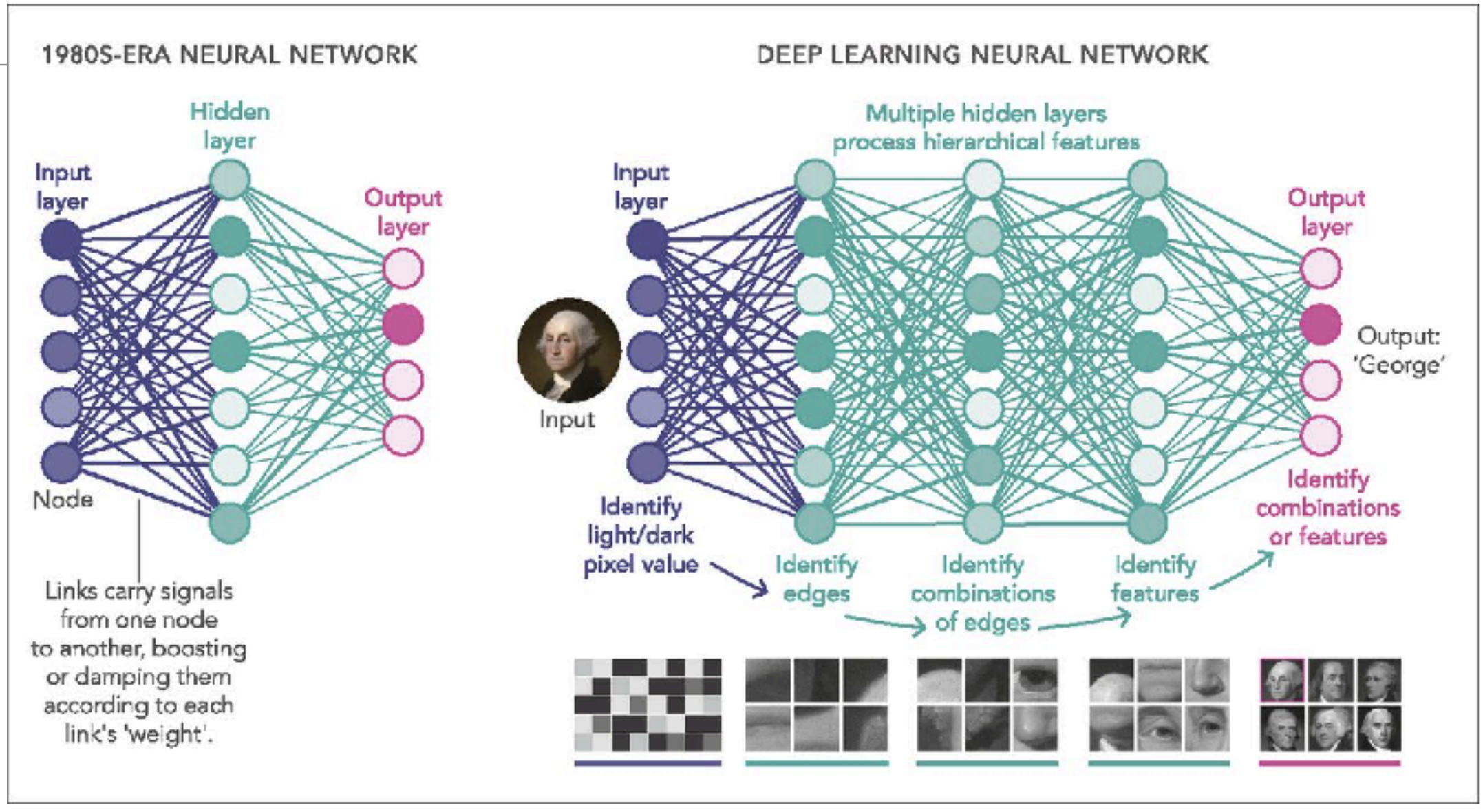



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Deep learning is  a class of machine learning algorithms that use multiple layers of artificial neural networks (ANNs) inspired by the structure and function of the human brain.

# 1.2 深度学习网络模型的结构

1980s　　2010s　　(From *PNAS* 116:1074-1077, 2019)



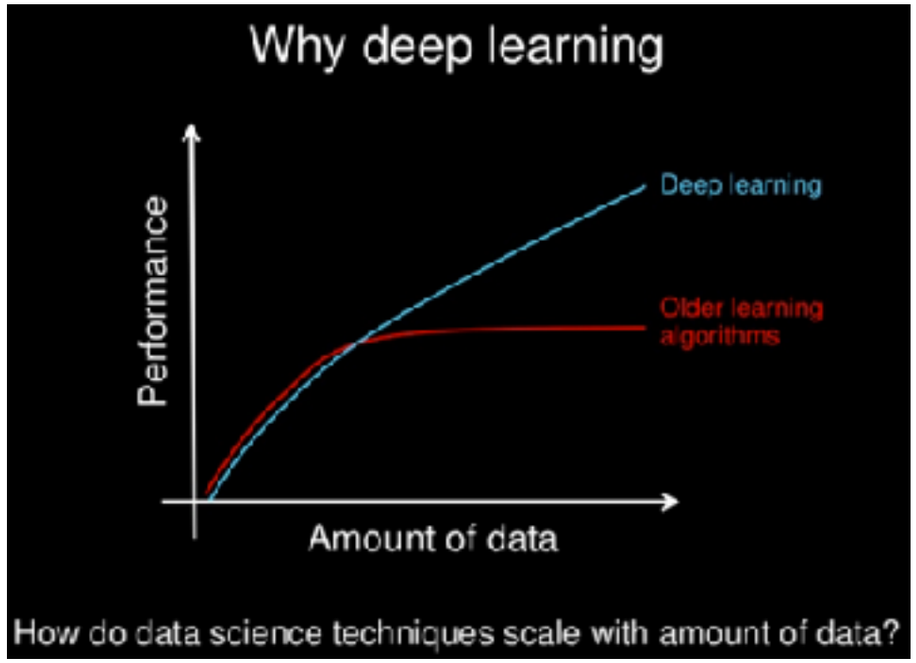

Shallow： one hidden layer　　Deep： many hidden layers

Deep learning uses multiple ANN layers to progressively extract higher level features from the raw input data.

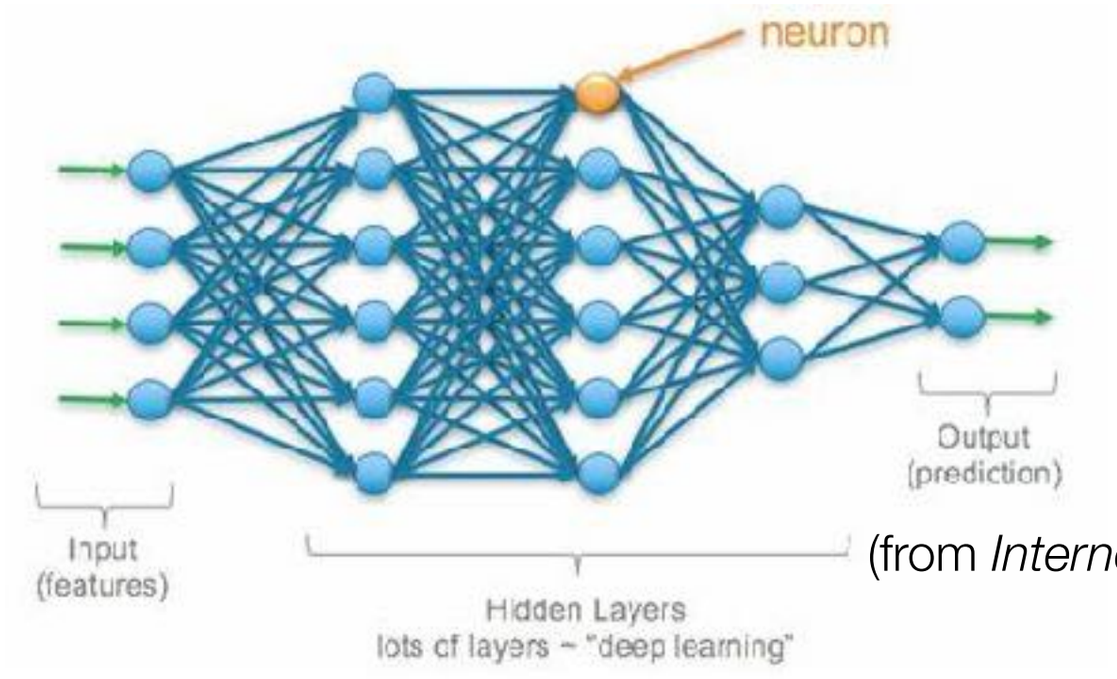
# 1.3 深度学习模型的基本特征

深度学习是一种自动化的、 具有多级表示的表征学习方法.

- Very large artificial neural networks.
- Huge amounts of data.
- More computation using bigger computers.




(From *Slide by Andrew Ng)*




(from *Internet*)

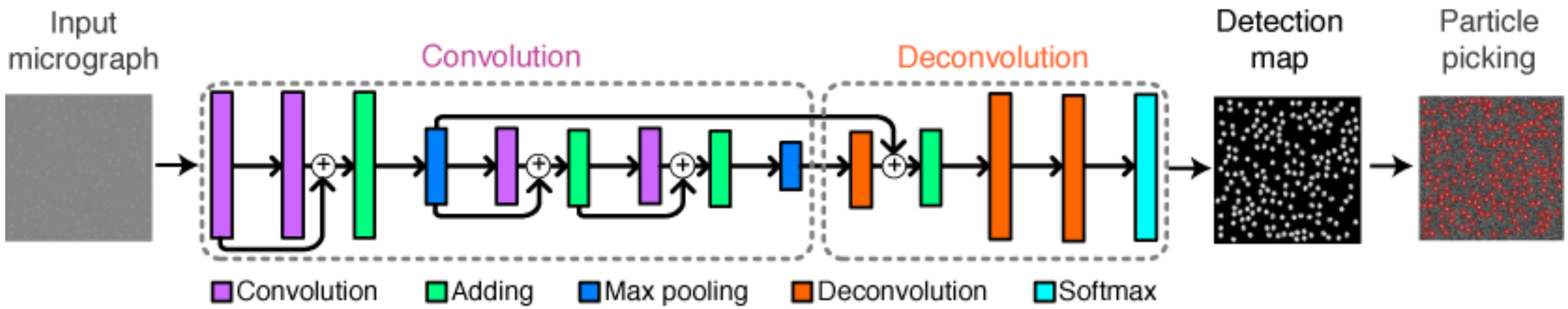

*Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.*

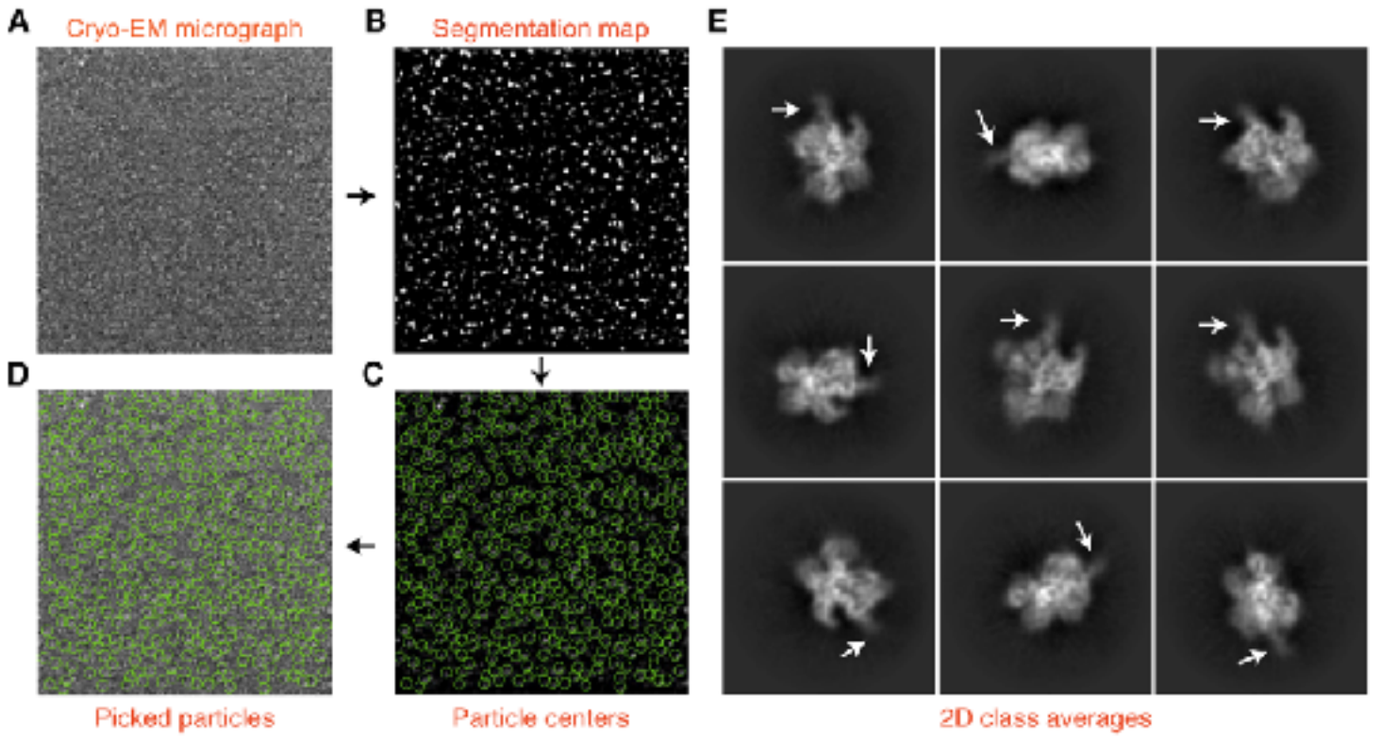
Yoshua Bengio

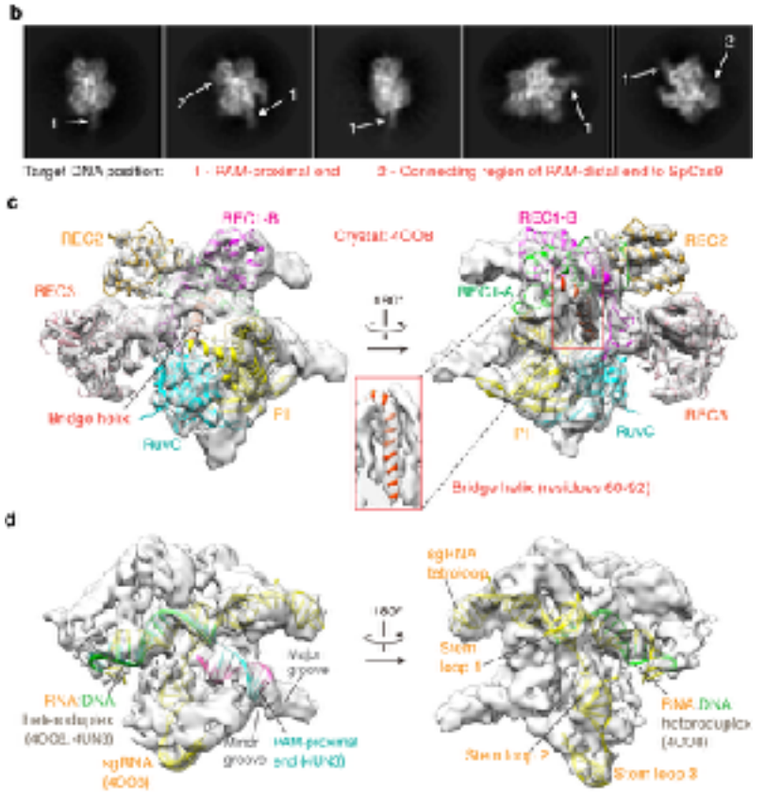
# 1.4 研究实例

Yao *et al.*, Deep learning with synthetic data enables automated picking of cryo-EM particle images of biological macromolecules. *Bioinformatics* (2019)

https://doi.org/10.1093/bioinformatics/btz728

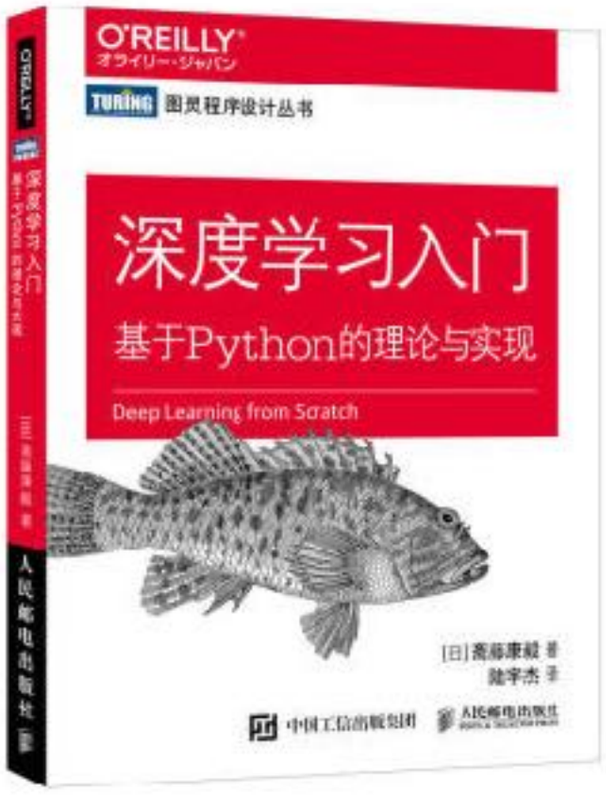


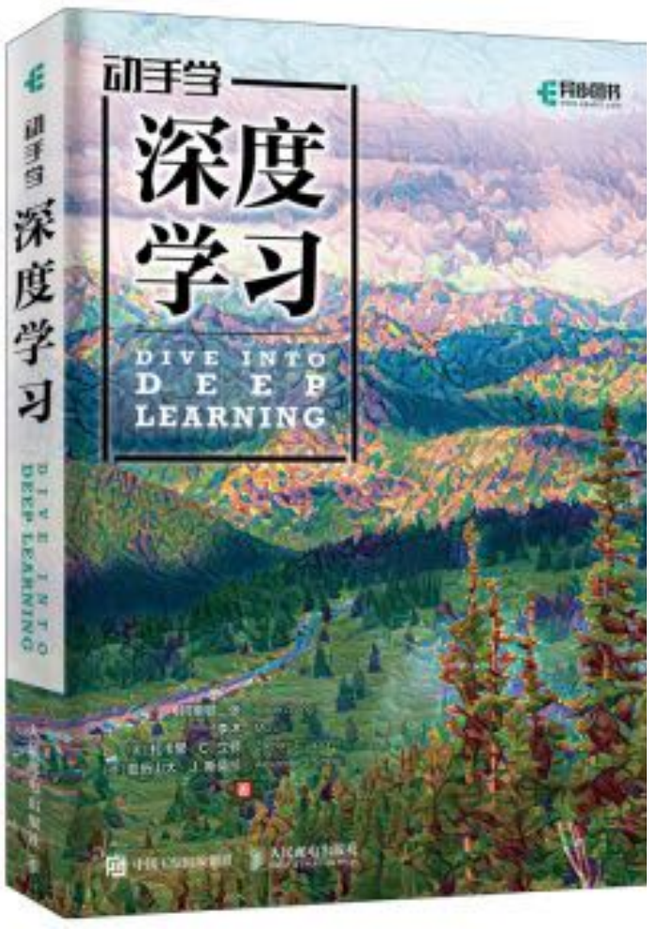



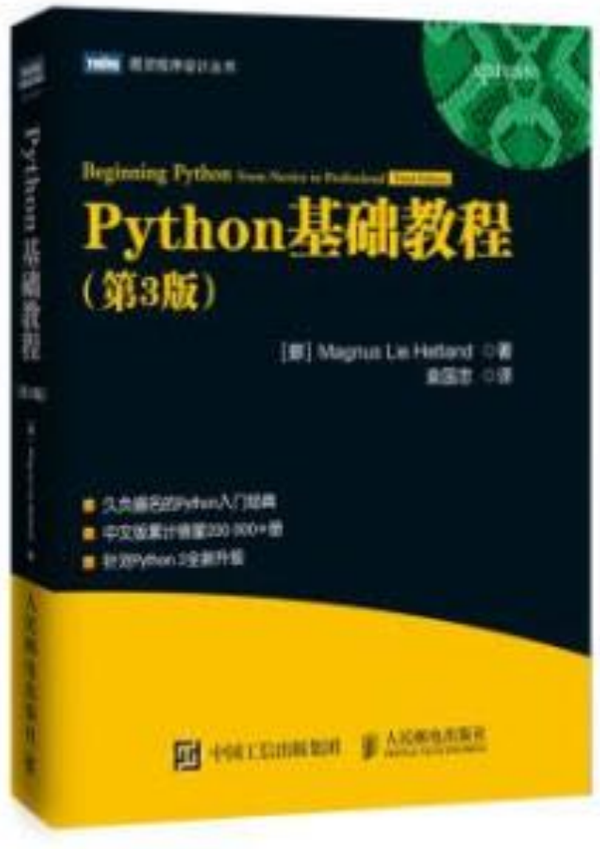



# 1.5 主要参考书

中国工信出版集团 人民邮电出版社




http://www.ituring.com.cn/book/1921 http://zh.d2l.ai http://ituring.com.cn/book/2118

请自行练习和实践参考书所带的Python程序！

# 本次课程内容

1 深度学习的基本概况

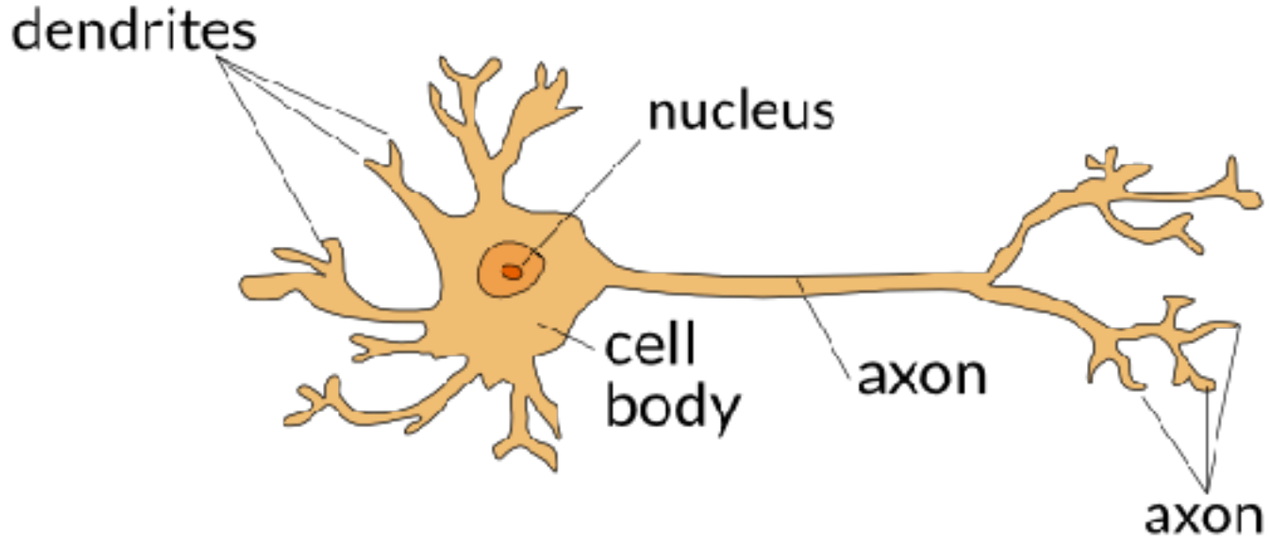
2 感知机模型与单层人工神经网络

3 多层神经网络与误差反向传播算法

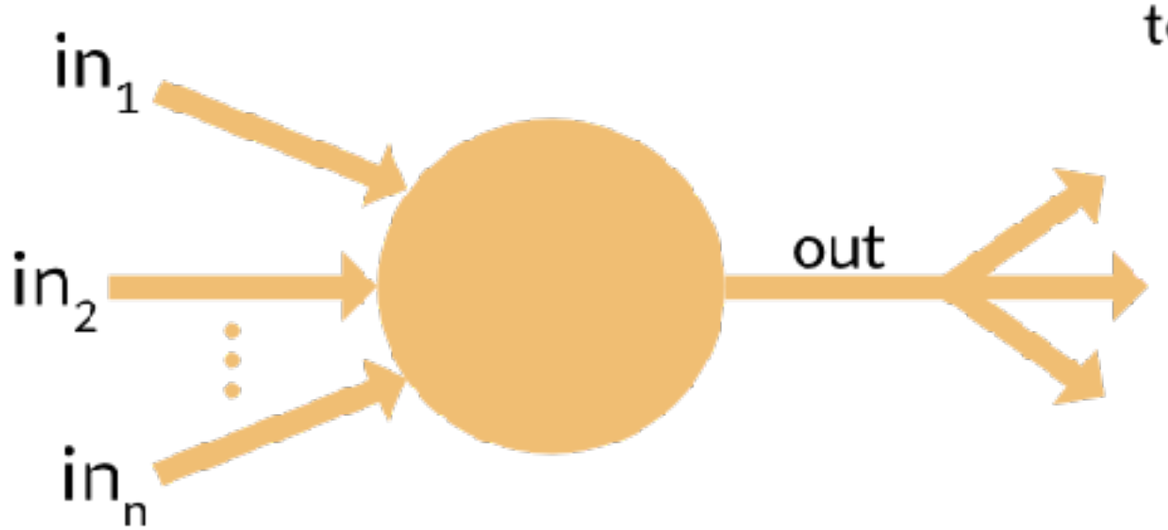
4 深度卷积神经网络与应用

# 2.1 神经元模型：感知机或感知器 (perceptron)

Biology
生物神经元




Model
人工神经元

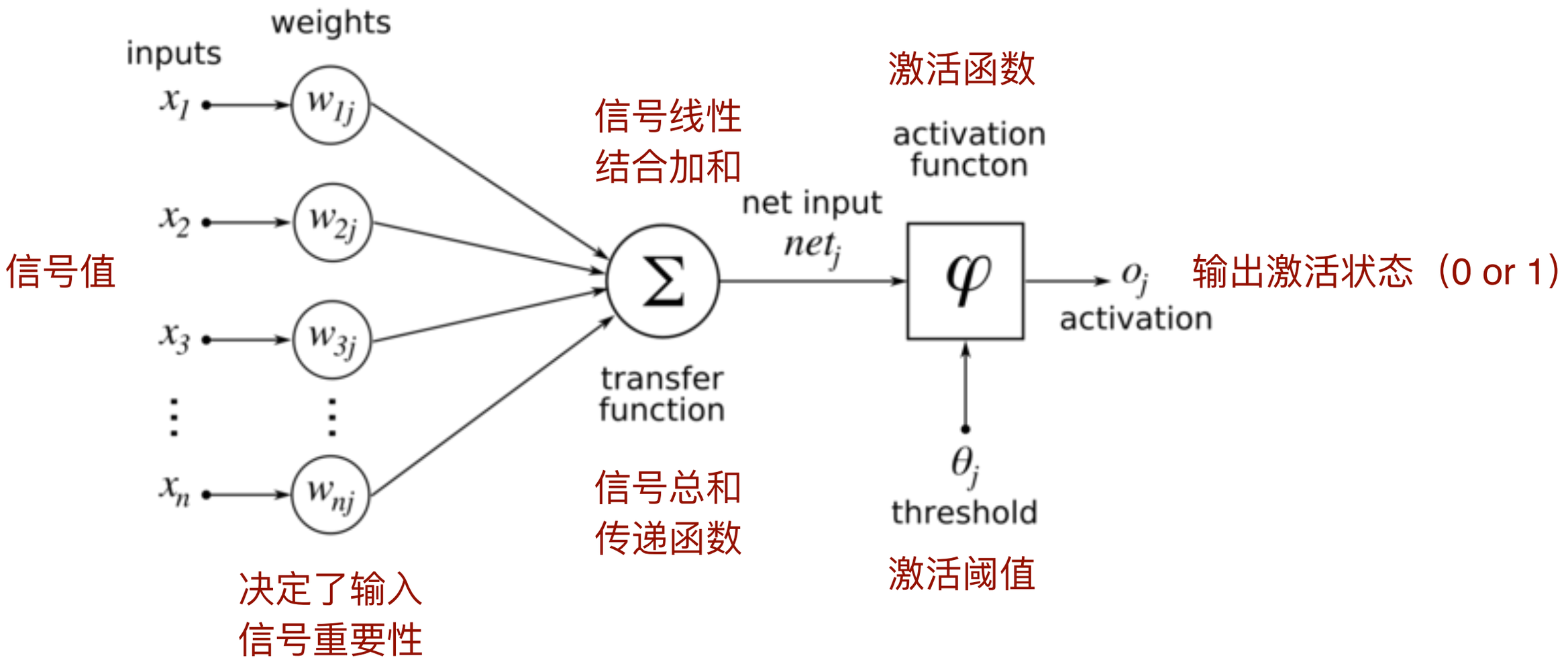



如生物神经元具有接收信号的树突、处理信号的细胞体、及将信号发送到其他神经元的轴突一样，人工神经元具有多个输入通道，信号处理单元和一个可以通向众多人工神经单元的输出通道。

# 2.2 人工神经元的数学模型

## 模仿生物神经元的激活过程

神经细胞的状态取决于从其它的神经细胞收到的输入信号量，及突触的强度（抑制或加强）。当信号量总和超过了某个阈值时，细胞体就会激动，产生电脉冲。电脉冲沿着轴突并通过突触传递到其它神经元。为了模拟神经细胞行为，与之对应的感知机基础概念被提出，如权重（突触）、偏置（阈值）及激活函数（细胞体）等。

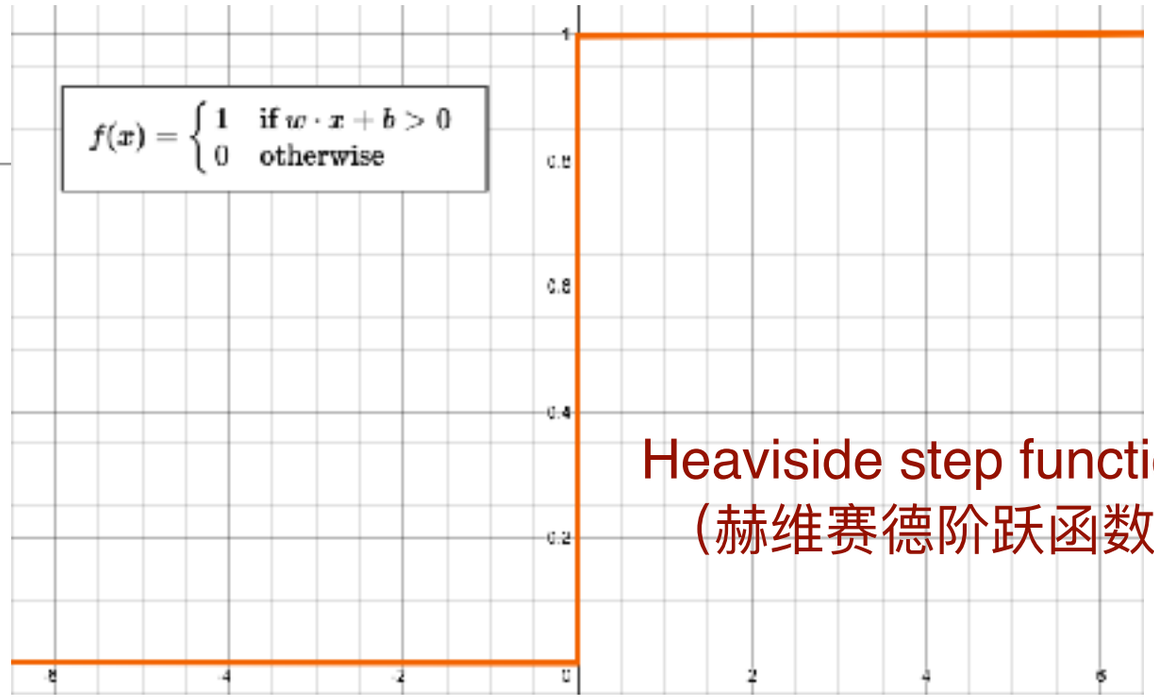



讨论与思考：如何实现数学表达式?

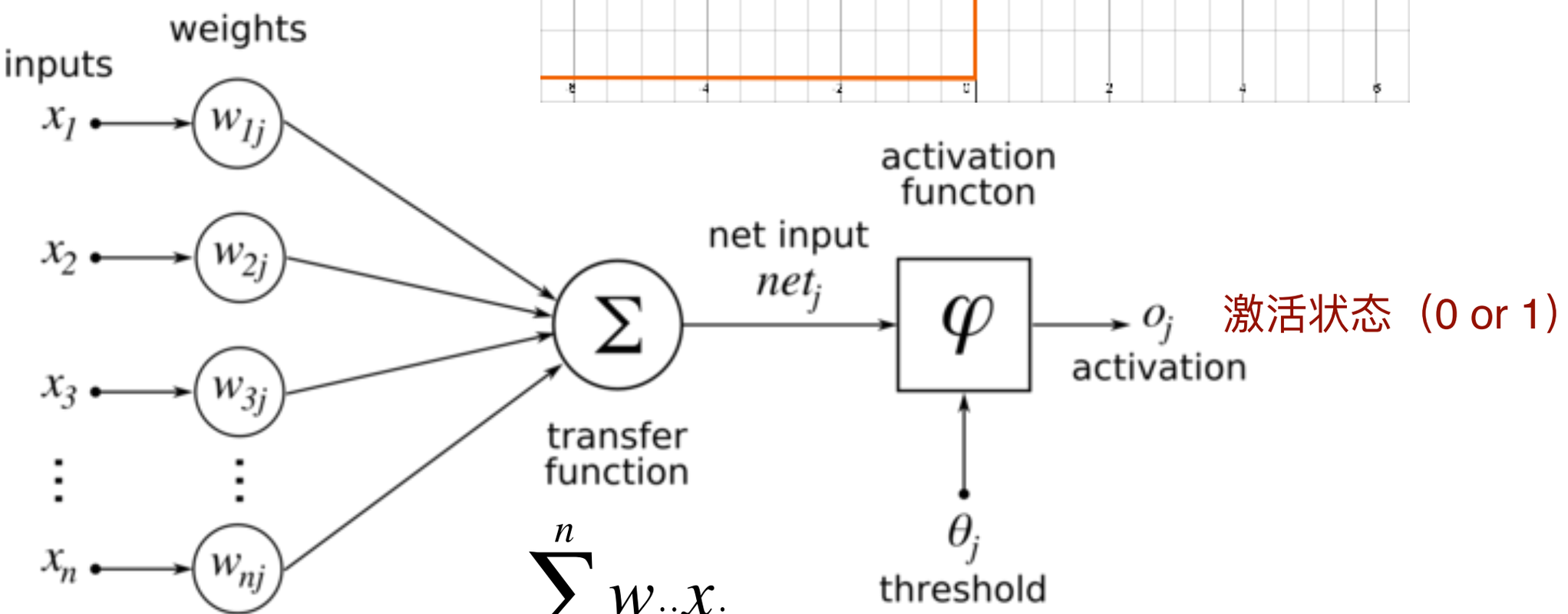
# 2.3 罗森布拉特（Rosenblatt's）感知机

## 人工神经元的激活函数

1957, 美国 Frank Rosenblatt 发明




Heaviside step function
（赫维赛德阶跃函数）




$$\sum_{i=1}^{n} w_{ij} x_i$$

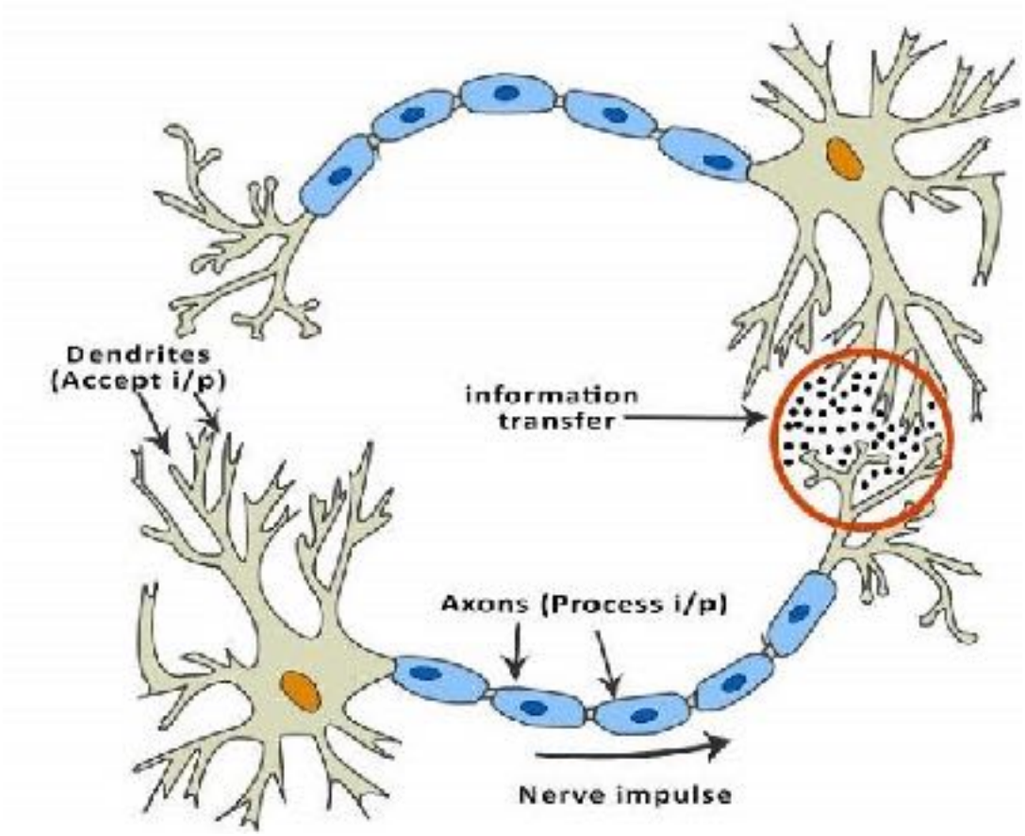
（向量内积： $\mathbf{W} \cdot \mathbf{X}$）

（或表示成： $\mathbf{W}^{\mathrm{T}}\mathbf{X}$ ）

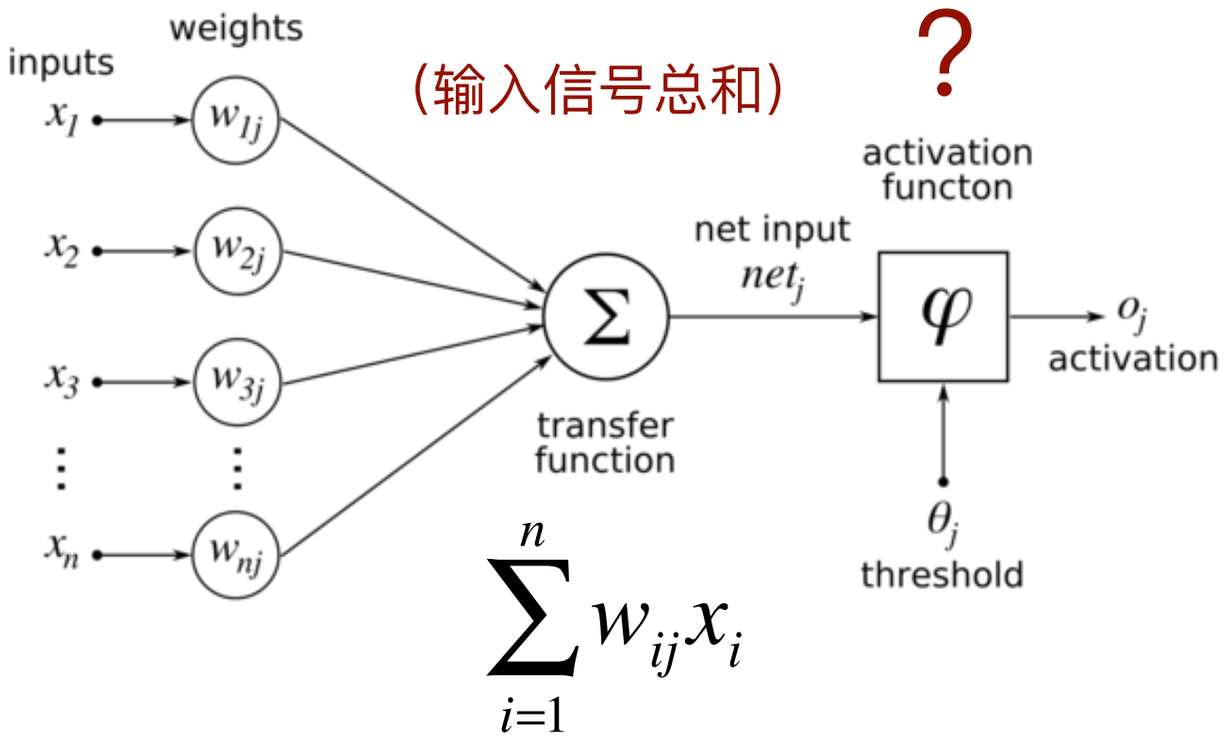
负值定义为偏置*b*， bias; 即平移；
等价于神经元的基础活跃等级

# 2.4 人工神经元的激活函数： 神经元的输入—输出变换

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. **These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.**




（activation process)

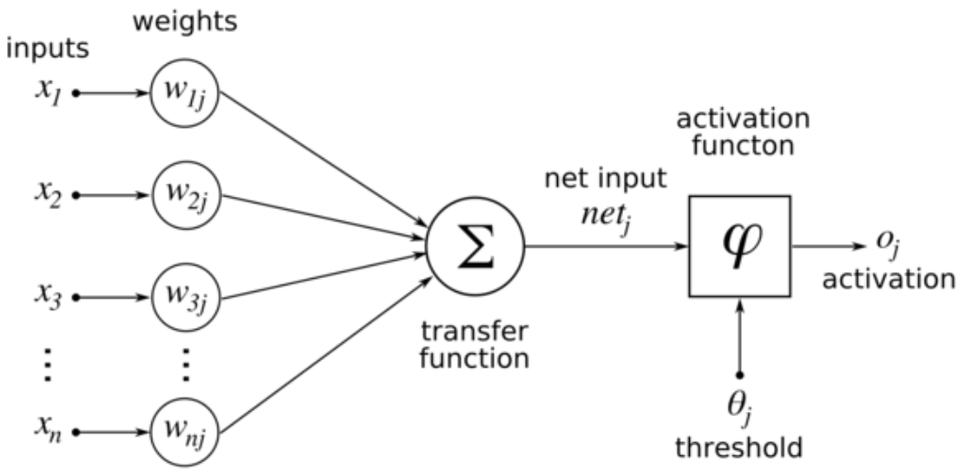


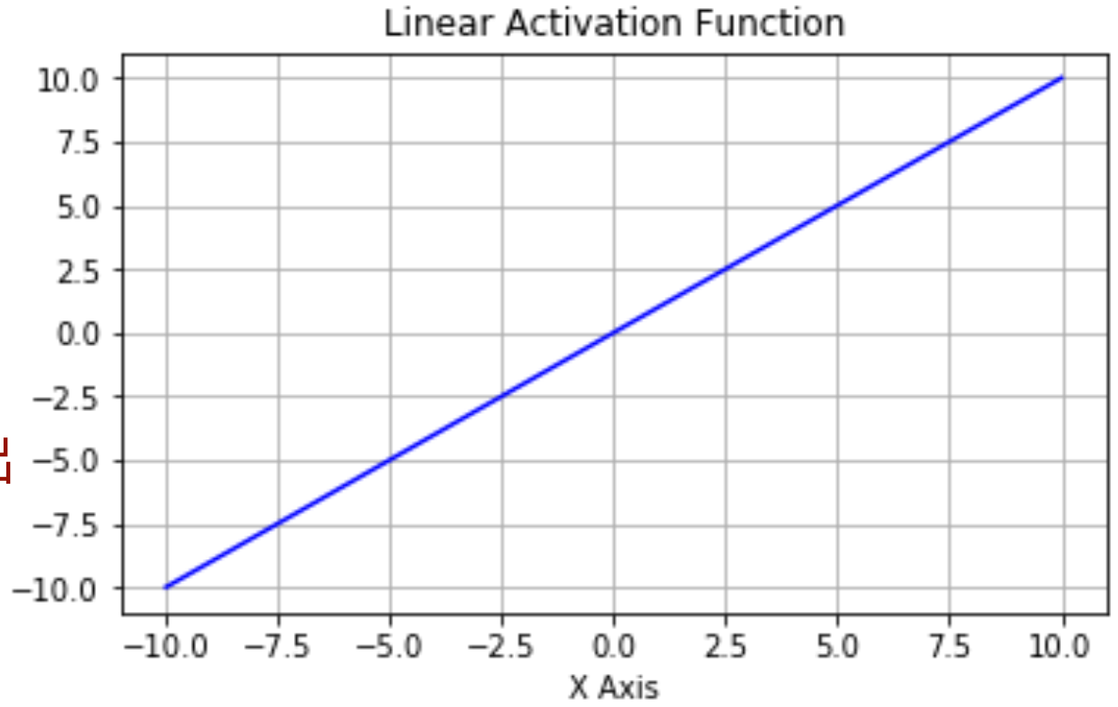

（线性组合） （线性或非线性）

通过激活函数可实现输入信号的非线性变换，决定了神经元(或节点)的输出。

# 2.5 线性激励函数 （Linear function)

输入信号总和不加处理，直接输出。



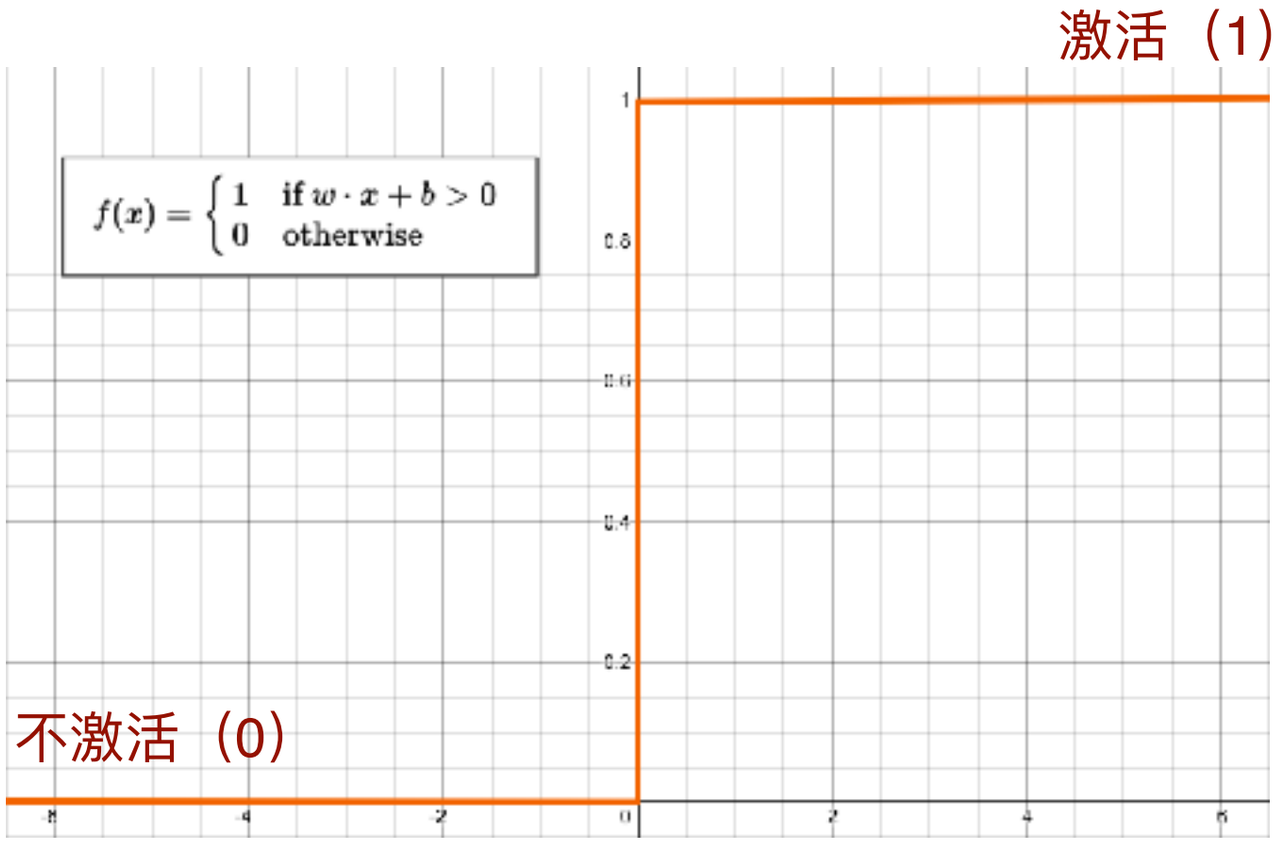




输出

输入 $\sum_{i=1}^{n} w_{ij} x_i$

# 2.6 阶跃函数 （Heaviside step function)

激活以阈值为界，当大于阈值，就切换输出。



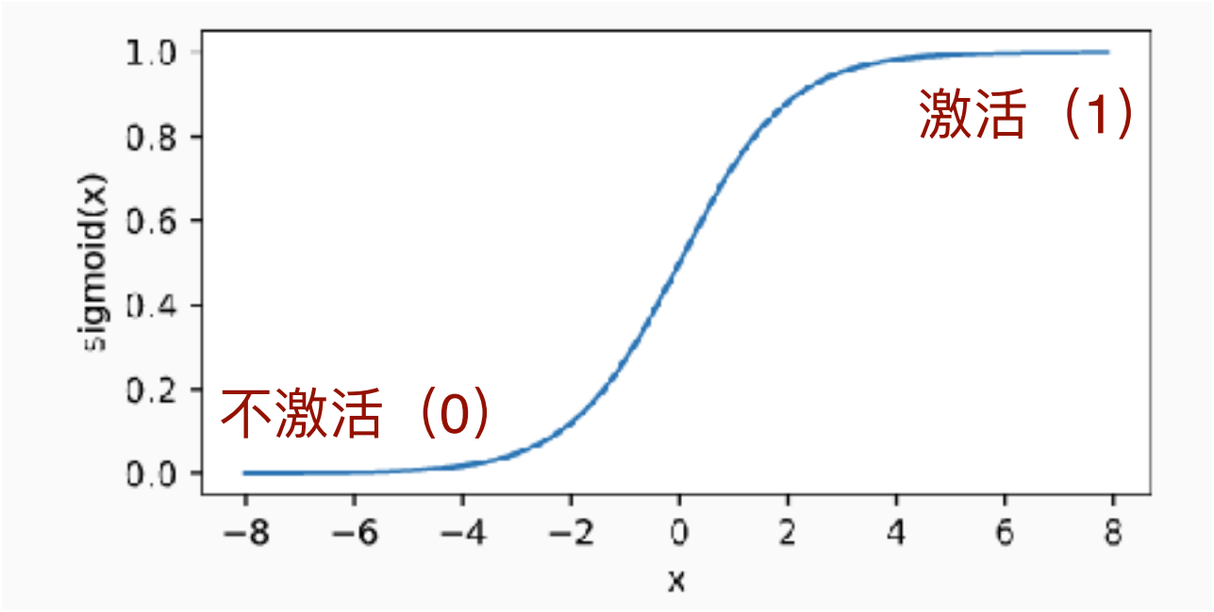

阈值：0

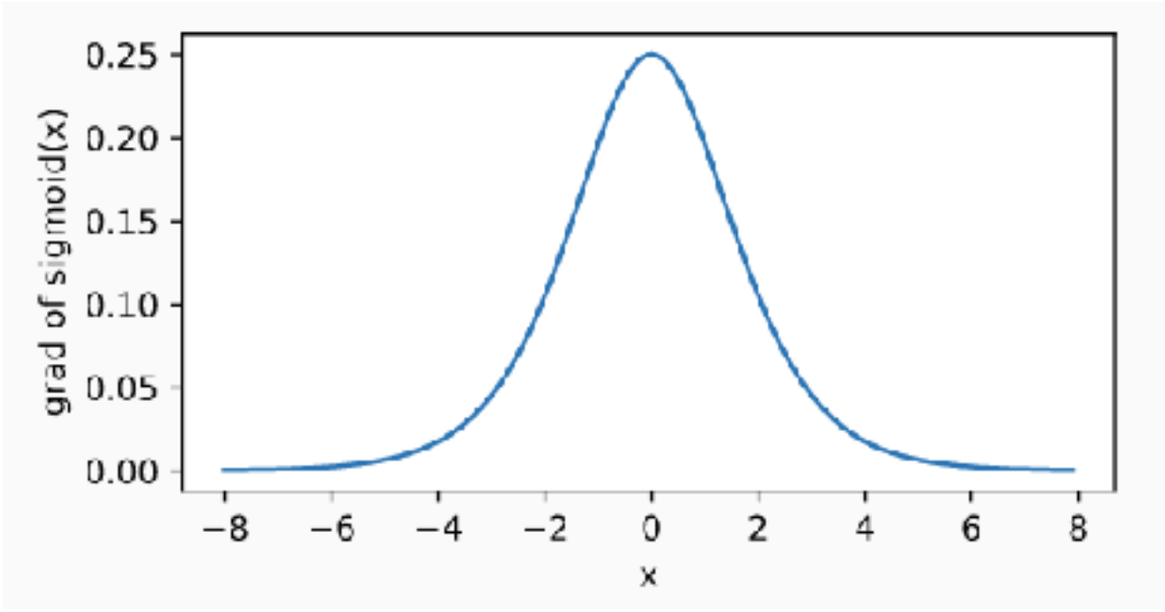
# 2.7 S形函数 （Sigmoid function）

输出随着输入发生连续性的变化。

$$\text{sigmoid}(x)=\frac{1}{1+\exp(-x)}$$

$$\text{sigmoid}'(x)=\text{sigmoid}(x)\left(1-\text{sigmoid}(x)\right)$$




当输入接近0时，sigmoid函数接近线性变换。

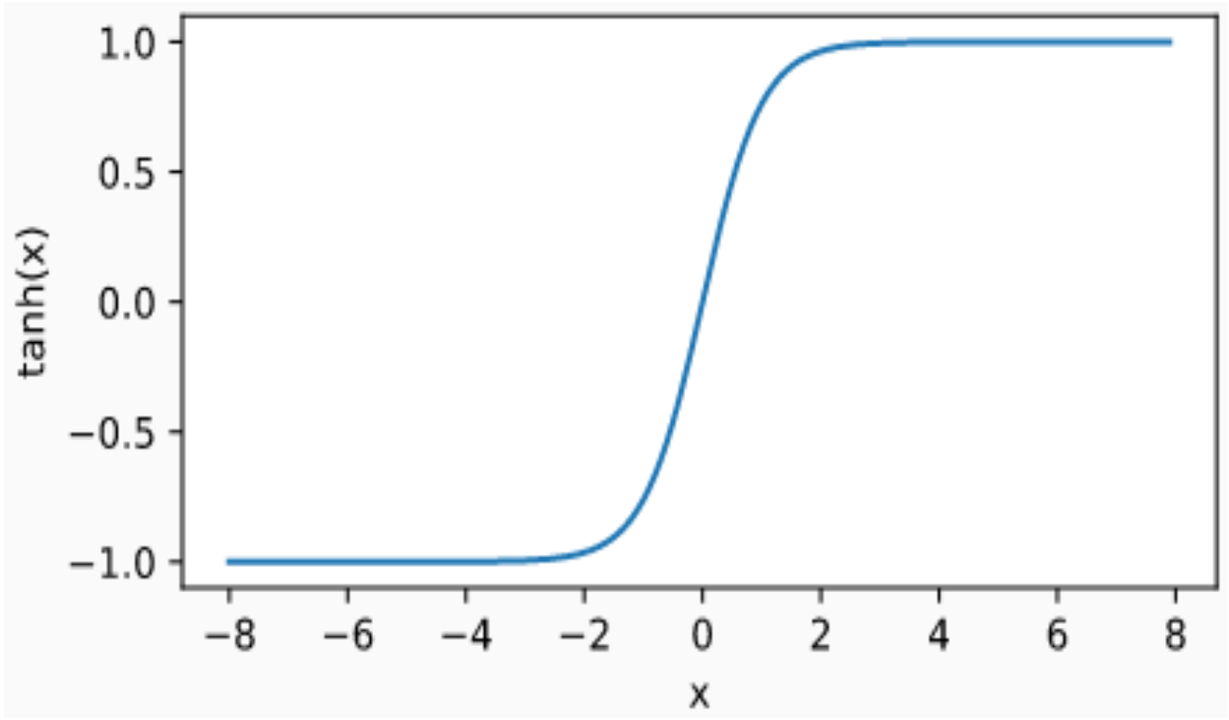
当输入为0时，sigmoid函数的导数达到最大值0.25；
当输入越偏离0时，sigmoid函数的导数越接近0。

# 2.8 双曲正切函数 （tanh function)

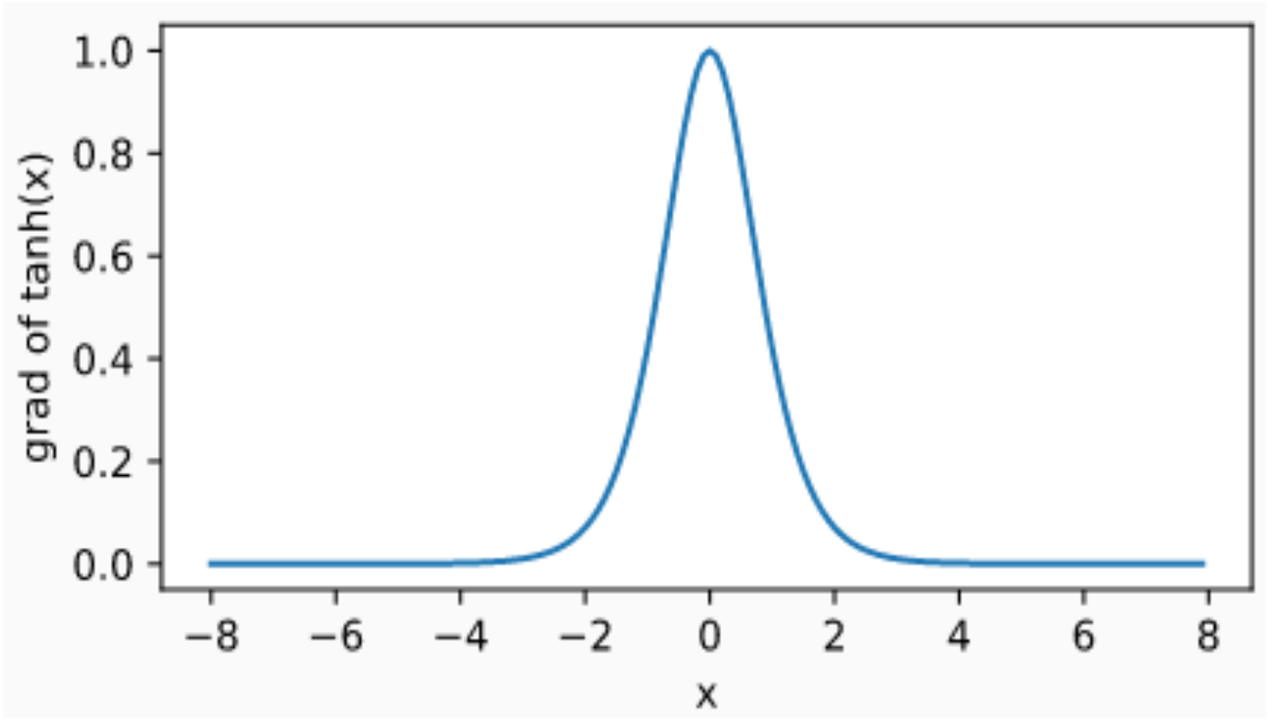
tanh（双曲正切）函数可以将元素的值变换到-1和1之间。

$$\tanh(x)=\frac{1-\exp(-2x)}{1+\exp(-2x)}$$

$$\tanh'(x)=1-\tanh^2(x)$$




tanh函数在坐标系的原点上对称。

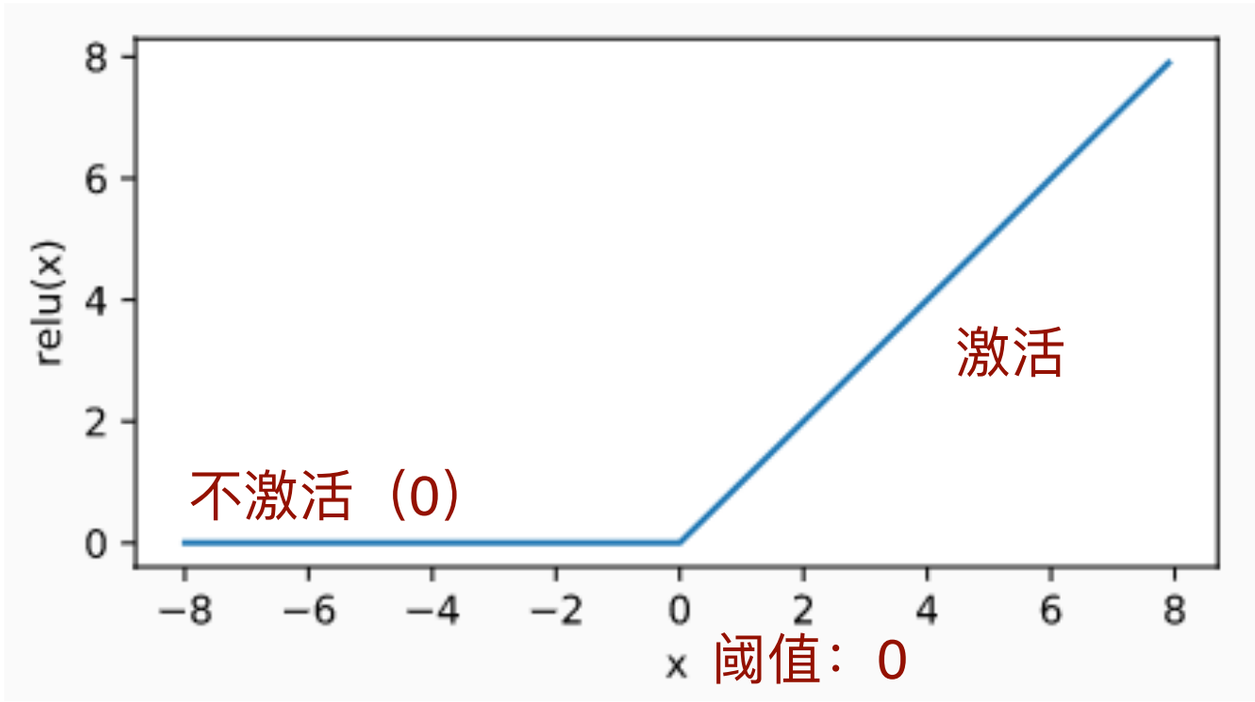



当输入为0时，tanh函数的导数达到最大值1。
当输入越偏离0时，tanh函数的导数越接近0。

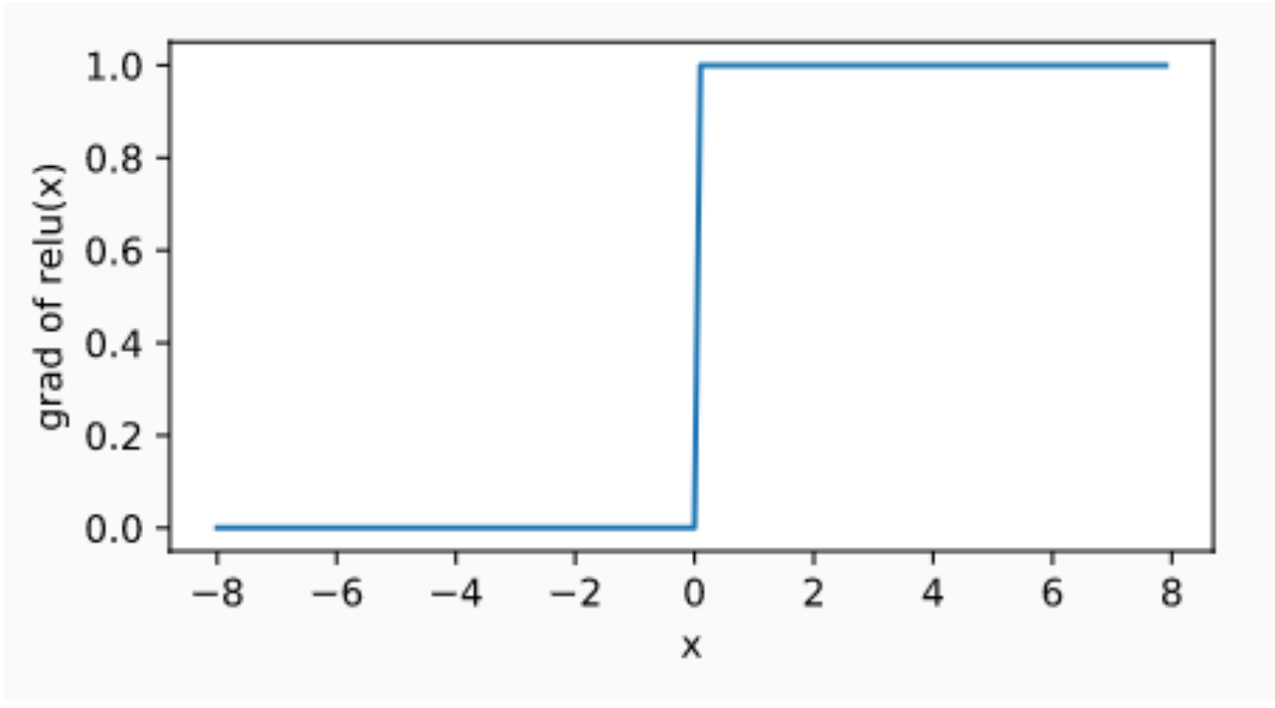
# 2.9 ReLU函数 （Rectified linear unit)

ReLU函数只保留正数元素，并将负数元素清零。

$$\mathrm{ReLU}(x) = \max(x, 0)$$



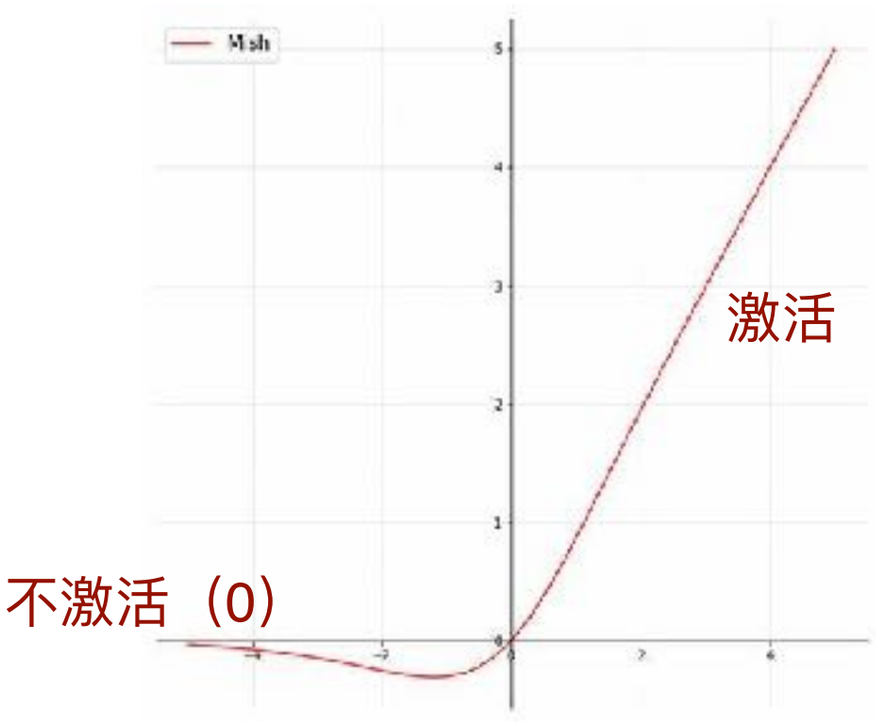

当输入为负数时，ReLU函数的导数为0；
当输入为正数时，ReLU函数的导数为1。



# 2.10 ReLU函数的新发展： Mish函数

ReLU函数在深度学习中占主导地位，但新函数还在不断发展。

$$f(x) = x \cdot \tanh[\ln(1 + e^x)]$$



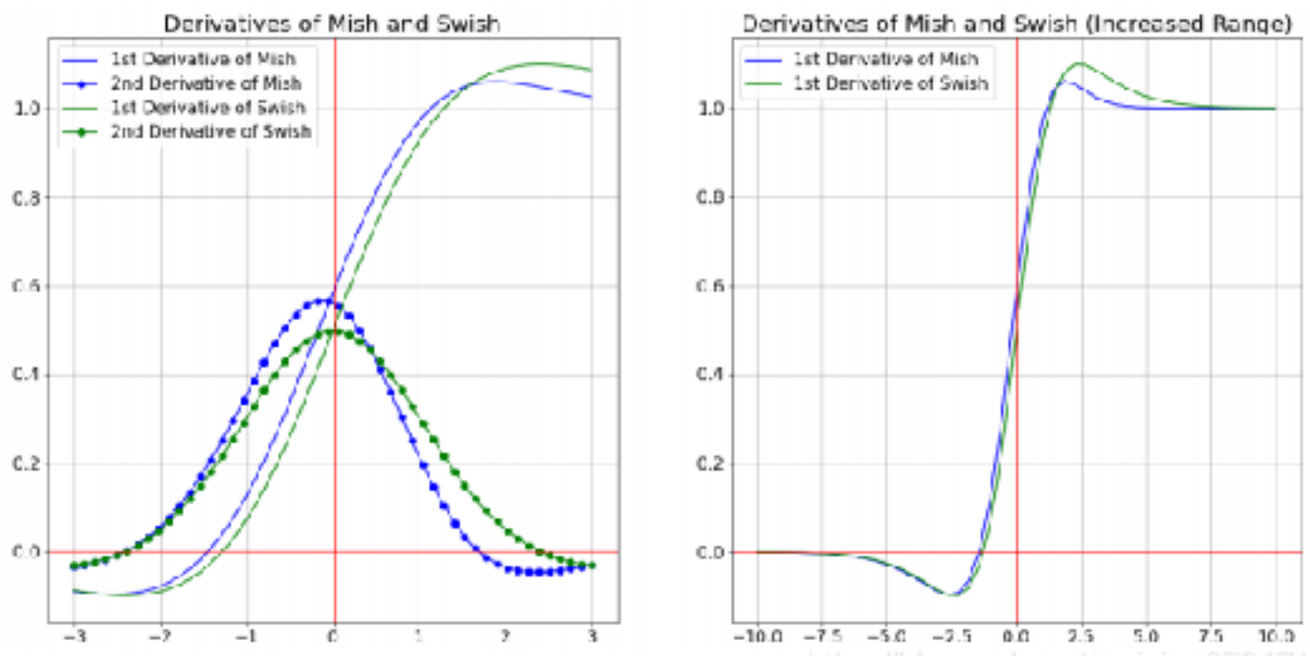




Figure 2. Comparison between 1st and 2nd derivative of Mish and Swish



Mish 是一种光滑非单调的函数，有下界，但无上界，取值范围是：[~-0.31, ∞]。

测试表明：Mish函数比ReLU函数有更好的结果。请参考：

https://arxiv.org/ftp/arxiv/papers/1908/1908.08681.pdf

# 2.11 Softmax函数 (归一化指数函数)

分类模型用，输出层含多个神经元（与类别数一致）。

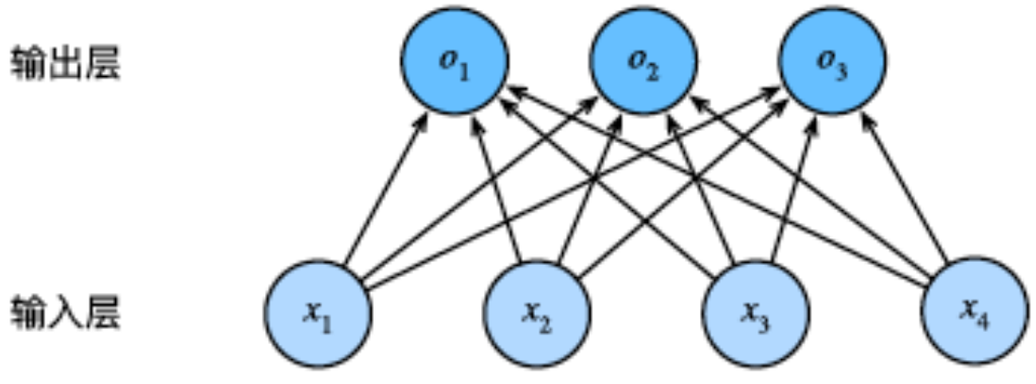
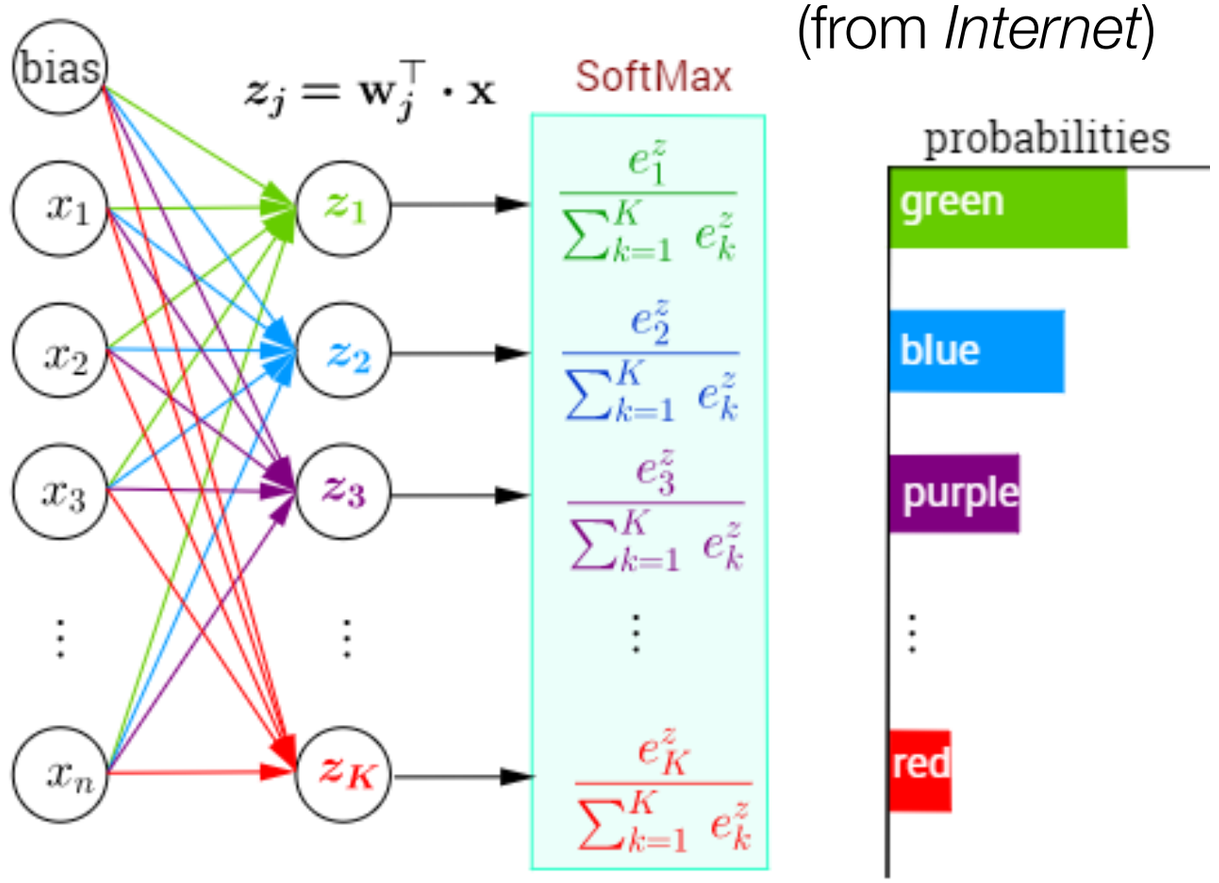



图 3.2: softmax回归是一个单层神经网络

(from *Internet*)




$$\hat{y}_1, \hat{y}_2, \hat{y}_3 = \text{softmax}(o_1, o_2, o_3),$$

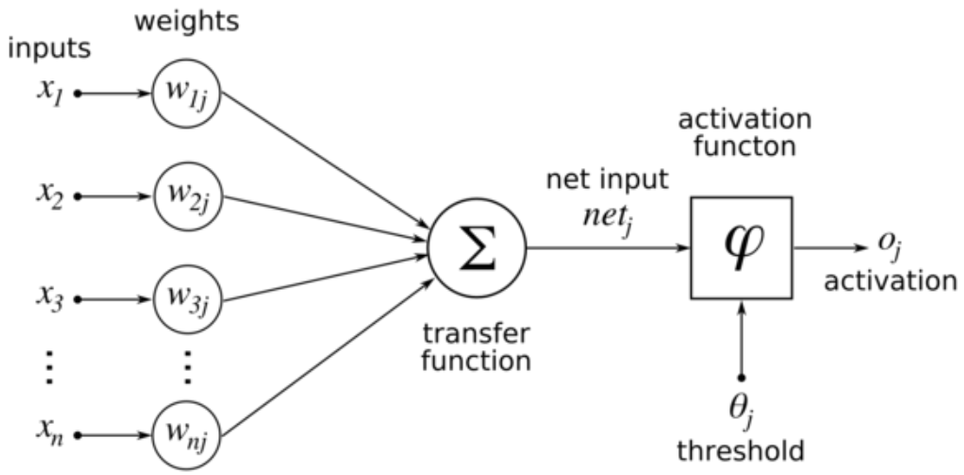
其中

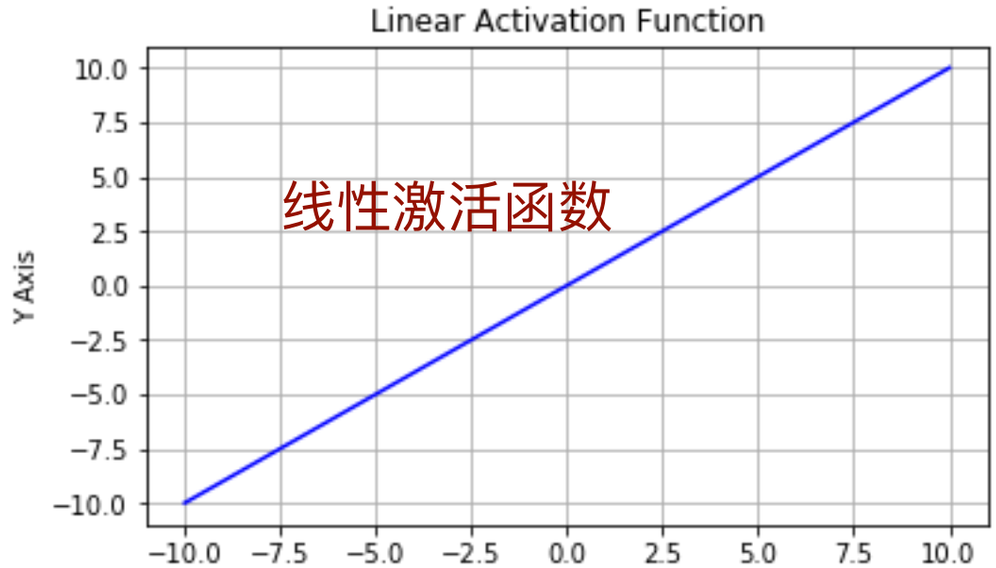
$$\hat{y}_1 = \frac{\exp(o_1)}{\sum_{i=1}^3 \exp(o_i)}, \quad \hat{y}_2 = \frac{\exp(o_2)}{\sum_{i=1}^3 \exp(o_i)}, \quad \hat{y}_3 = \frac{\exp(o_3)}{\sum_{i=1}^3 \exp(o_i)}.$$

容易看出$\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$且$0 \leq \hat{y}_1, \hat{y}_2, \hat{y}_3 \leq 1$，因此$\hat{y}_1, \hat{y}_2, \hat{y}_3$是一个合法的概率分布。

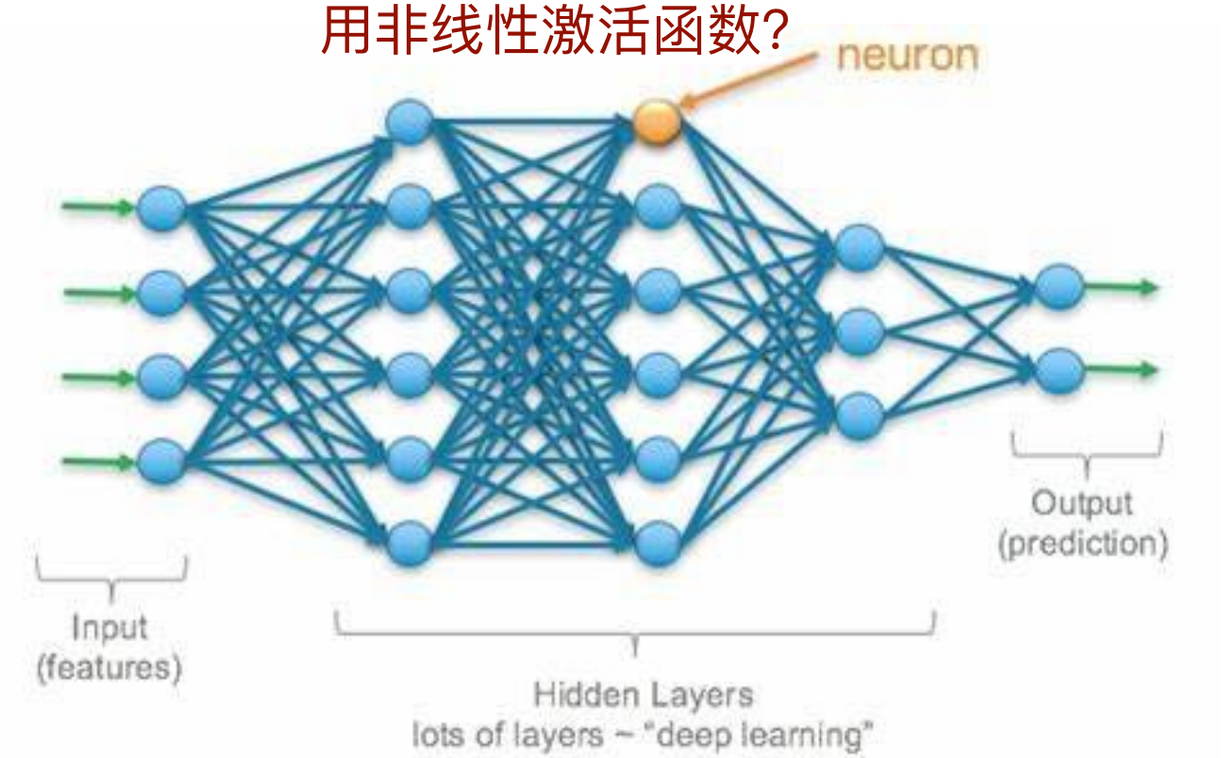
# 2.12 课后思考

为什么多层神经网络模型的激活函数需要使用非线性函数?



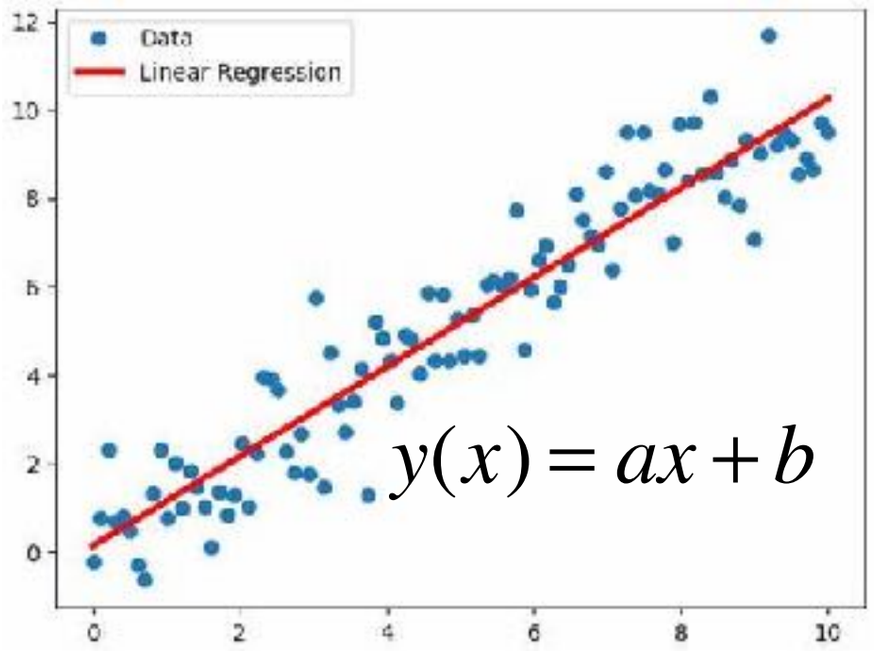

用非线性激活函数?

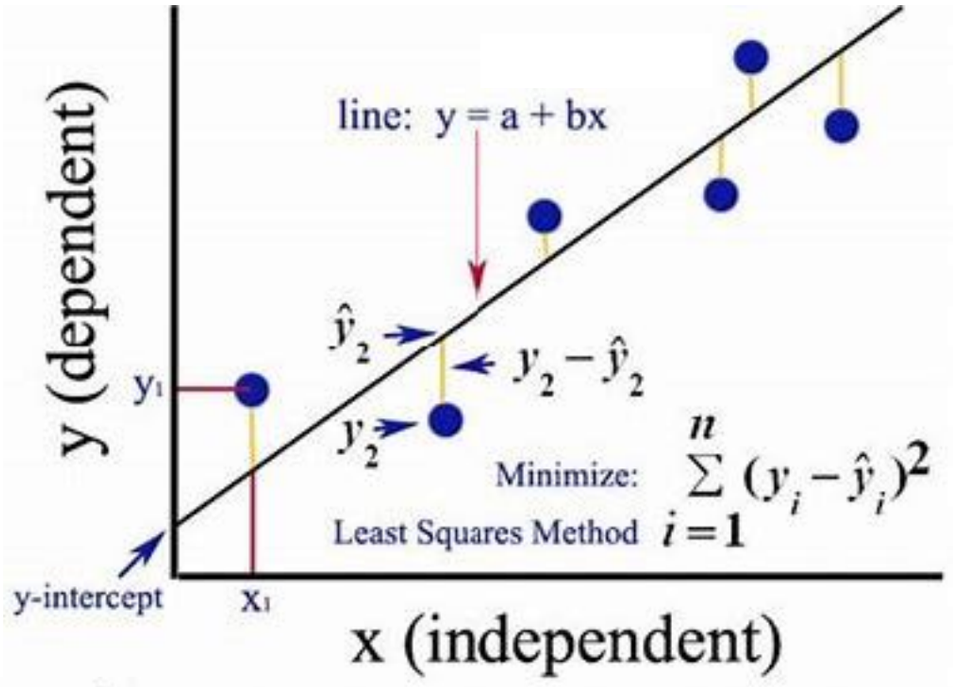


# 2.13 知识回顾：一元或多元线性回归

一元线性回归模型：$y(x) = ax + b$

问题：如果根据n个样本点求得模型参数*a* 和 *b*？ 精确的解析解？请自己推导公式。



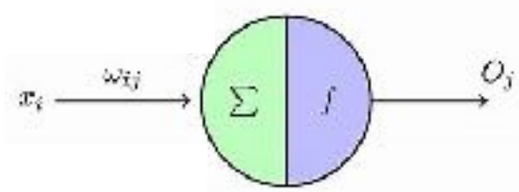

最小二乘拟合法获得的*a* 和 *b*解析解：

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$b = \frac{1}{n}(\sum_{i=1}^{n} y_i - a\sum_{i=1}^{n} x_i)$$

思考：方法核心？

可用单层神经网络类比




线性激活函数

# 2.14 方法应用：用单层神经网络模型实现多元线性回归

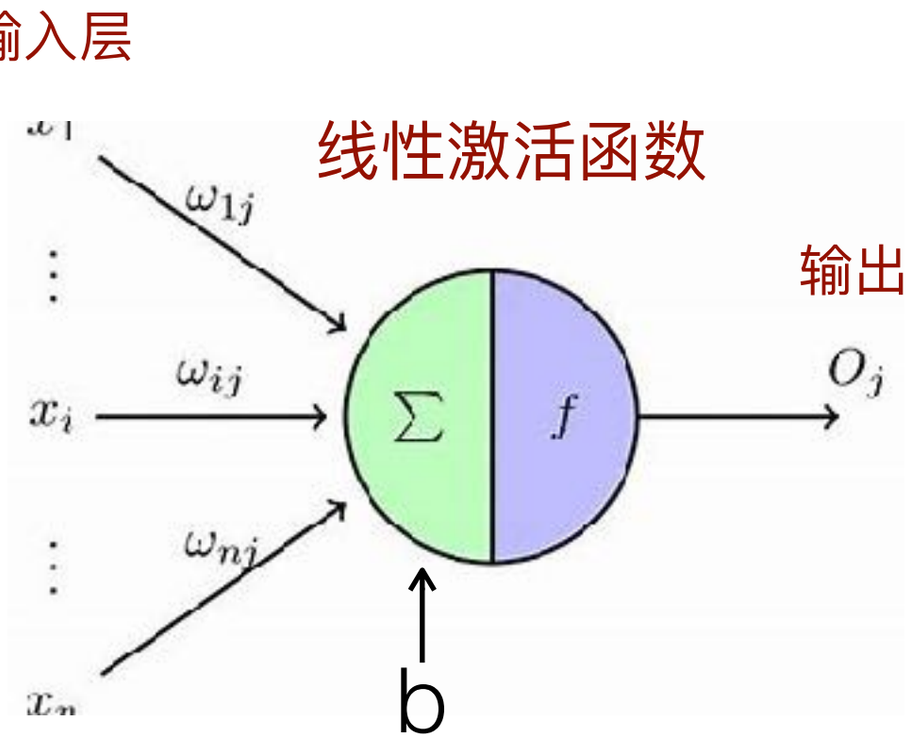
第一步：设置神经网络模型

多元线性回归（multiple linear regression) 可以用单层神经网络模型表示

$$y(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

| 序号 | 对某商品的消费支出Y | 商品单价 $x_1$ | 家庭月收入 $x_2$ |
|---|---|---|---|
| 1 | 591.9 | 23.56 | 7620 |
| 2 | 654.5 | 24.44 | 9120 |
| 3 | 623.6 | 32.07 | 10670 |
| 4 | 647.0 | 32.46 | 11160 |
| 5 | 674.0 | 31.15 | 11900 |
| 6 | 644.4 | 34.14 | 12920 |
| 7 | 680.0 | 35.30 | 14340 |
| 8 | 724.0 | 38.70 | 15960 |
| 9 | 757.1 | 39.63 | 18000 |
| 10 | 706.8 | 46.68 | 19300 |

样本数据 (示例)




n个输入的单层神经网络模型

问题：如果根据样本数据集求得参数：$w_1, w_2, b$ ？

思考：如何“训练”神经网络模型（即求出合适的权重与偏置值）？

# 2.15 损失函数（Loss function）：使平方误差函数最小化

第二步：定义神经网络模型的计算输出值与样本实际值（或实验值，或观察值）的误差函数

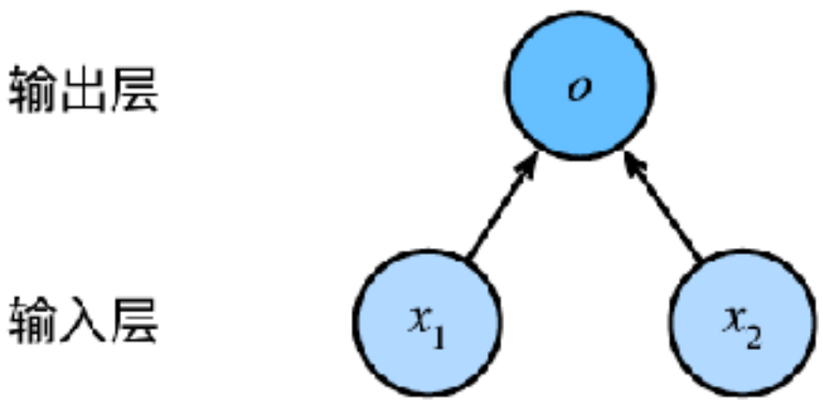



图 3.1: 线性回归是一个单层神经网络

$$y(x_1,x_2)=w_1x_1+w_2x_2+b$$

计算值　实验值

样本 *i* 的平方误差

$$\ell^{(i)}(w_1,w_2,b)=\frac{1}{2}\left(\hat{y}^{(i)}-y^{(i)}\right)^2,$$

平方误差函数

全部样本的平方误差

$$\ell(w_1,w_2,b)=\frac{1}{n}\sum_{i=1}^{n}\ell^{(i)}(w_1,w_2,b)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(x_1^{(i)}w_1+x_2^{(i)}w_2+b-y^{(i)}\right)^2.$$

网络训练目标：

$$w_1^*,w_2^*,b^*=\underset{w_1,w_2,b}{\operatorname{argmin}}\ \ell(w_1,w_2,b).$$

（求得使损失函数最小化的参数集！）

（神经网络参数为损失函数的自变量！）

# 2.16 模型参数的数值最优化算法

第三步：用迭代算法训练神经网络模型，自动从样本数据求得合适的参数
（小批量随机梯度下降　Mini-batch stochastic gradient descent，SGD)

1. 先设定一组模型参数的初始值，如随机选取；

2. 对参数进行多次迭代修正，使每次迭代都可能降低损失函数的值：

误差函数对参数的梯度　　学习(速)率：人为预先设定正数

$$w_1 \leftarrow w_1 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1,w_2,b)}{\partial w_1} = w_1 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} x_1^{(i)}\left(x_1^{(i)}w_1 + x_2^{(i)}w_2 + b - y^{(i)}\right),$$

损失函数对参数w1的偏导数

$$w_2 \leftarrow w_2 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1,w_2,b)}{\partial w_2} = w_2 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} x_2^{(i)}\left(x_1^{(i)}w_1 + x_2^{(i)}w_2 + b - y^{(i)}\right),$$

$$b \leftarrow b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1,w_2,b)}{\partial b} = b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\left(x_1^{(i)}w_1 + x_2^{(i)}w_2 + b - y^{(i)}\right).$$

小批量（mini-batch） $\mathcal{B}$

每次迭代，可先随机均匀采样一个由固定数目训练数据样本所组成的小批量(mini-batch) $\mathcal{B}$，然后求小批量中数据样本的平均损失有关模型参数的偏导数（梯度），最后用此结果与预先设定的一个正数的乘积作为模型参数在本次迭代的减小量。

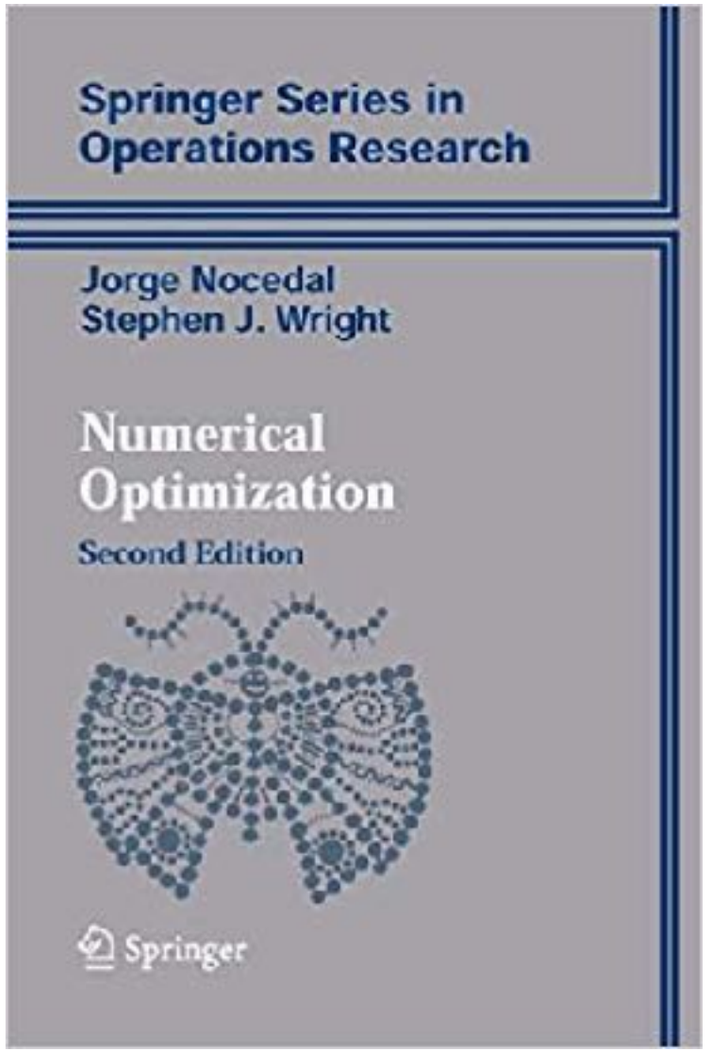
3. 当损失函数不再降低，停止迭代计算，输出模型参数。

深入思考：为什么每一次迭代都用“梯度”来修正?

# 知识拓展：梯度下降等最优化算法的参考书

## 科学出版社


## Springer出版社




计算数学中的 **梯度下降法** 是神经网络模型参数训练的核心最优化算法。