

# 机器学习基本概念

---

机器学习~=找一个（人类写不出来的）函数，而深度学习是利用**类神经网络**找函数

输入：向量/矩阵/序列（语音文字） -> 输出：数值（回归regress） / 类别（分类） / 图像文本

## 机器学习任务

- 如何教：
  - Supervised Learning：有数据集，有标签 data and label
  - Self-supervised Learning：利用unlabeled数据集学一些基本任务Pre-Train（Pre-trained Model又称为Foundation Model e.g.: Bert）
  - Generative Adversarial Network：不需要数据和标签成对
  - Reinforcement Learning：不知道最好的选择是啥
- 关注点：
  - Anomaly Detection：异常检测
  - Explainable AI：可解释AI，讲为什么
  - Model Attack：模型攻击
  - Domain Adaptation：自适应
  - Network Compress：模型压缩
  - Life-long Learning
- 学习如何学习
  - Meta learning / Few-shot learning：机器自己从数据中发明一种算法，从少量数据学习。

## 深度学习概念推导

(1) 步骤：

- 设计Model  $y = b + wx_1$   $w$ 和 $b$ 需要机器学习
- 从训练数据中设计Loss 损失是参数的函数 $L(b, w)$  估值和标签的差距

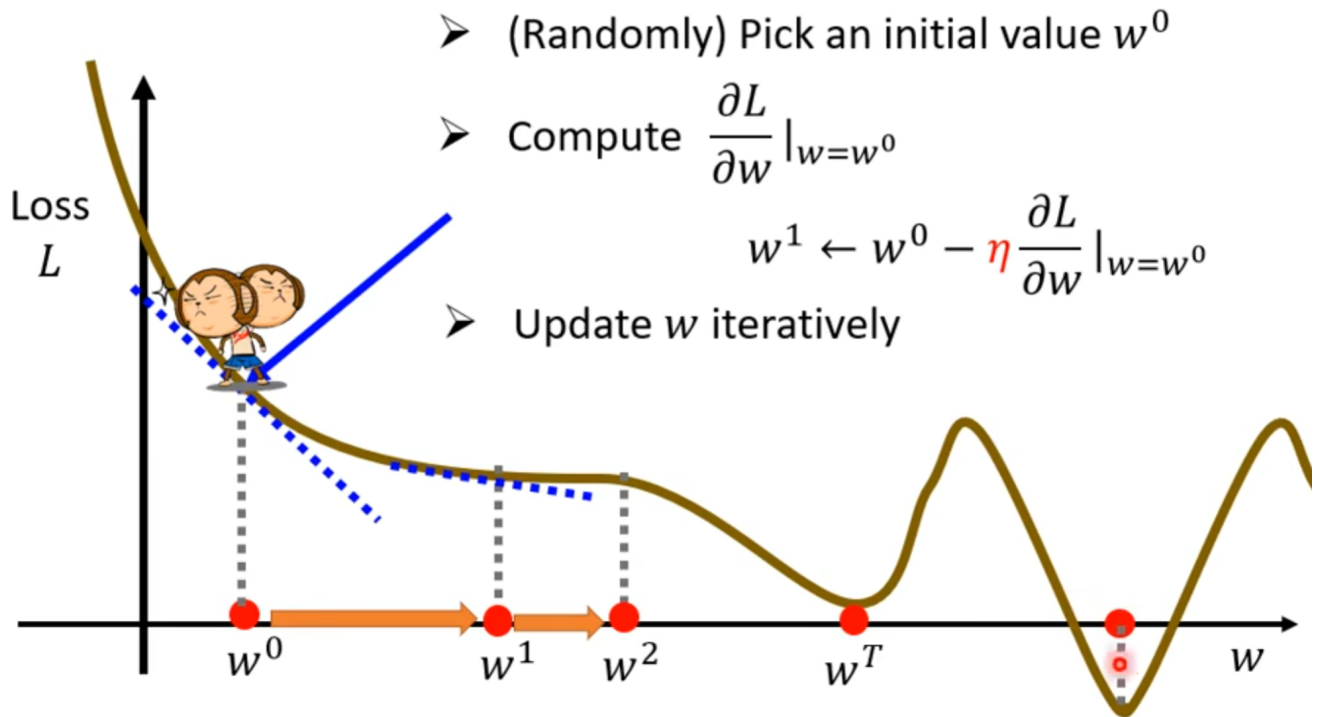
- Optimization最佳化:

每次遮住只保留一个未知数（其实就是对损失函数求偏导）。

### 3. Optimization

$$w^* = \arg \min_w L$$

#### Gradient Descent



如果算的是负数，说明左大右小，增加 $w$ 。学习率自己设置 $\eta$ ，影响偏移的范围

### 3. Optimization

$$w^*, b^* = \arg \min_{w, b} L$$

➤ (Randomly) Pick initial values  $w^0, b^0$

➤ Compute

$$\begin{aligned} \frac{\partial L}{\partial w} \big|_{w=w^0, b=b^0} \\ \frac{\partial L}{\partial b} \big|_{w=w^0, b=b^0} \end{aligned}$$



$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w} \big|_{w=w^0, b=b^0}$$


$$b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b} \big|_{w=w^0, b=b^0}$$

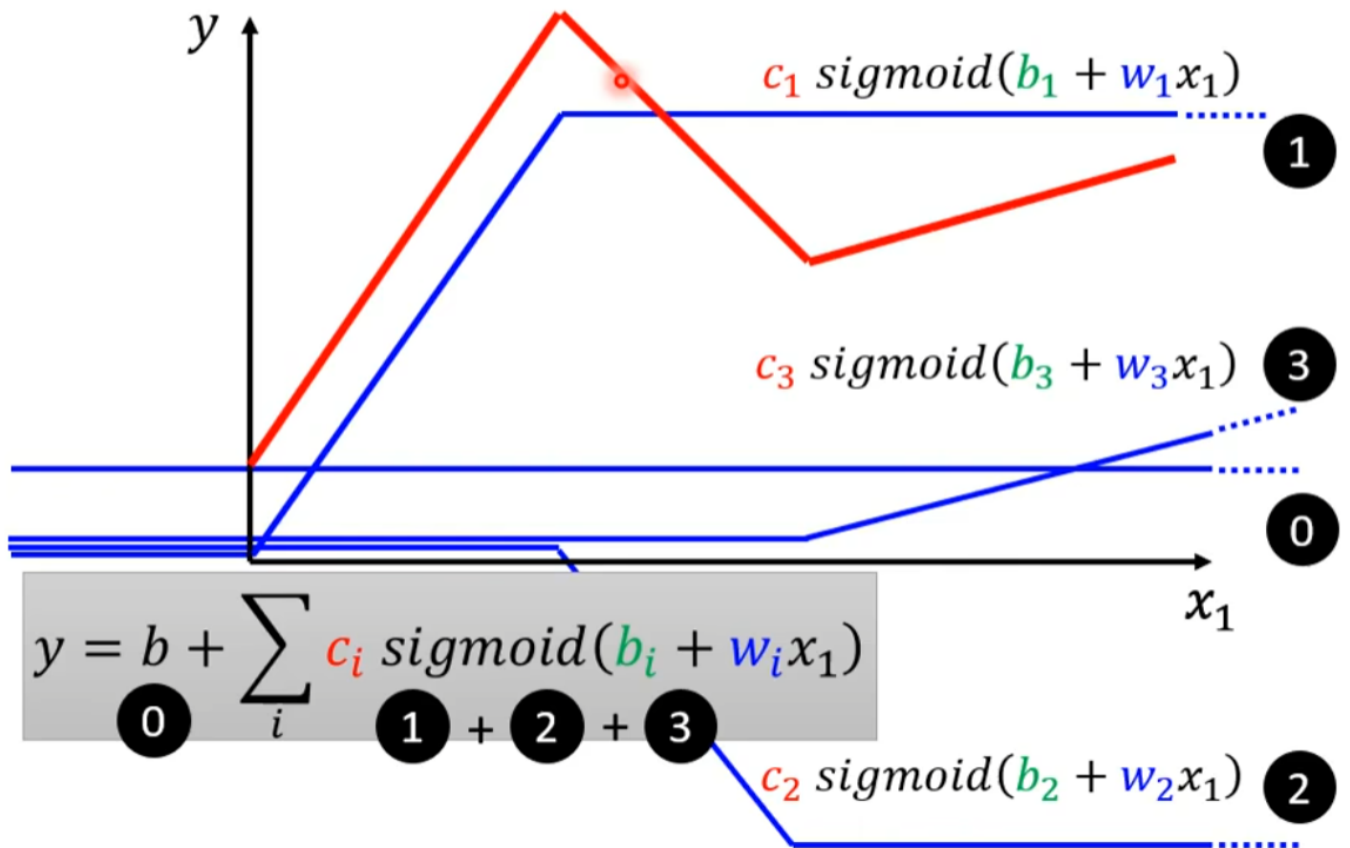
Can be done in one line in most deep learning frameworks

➤ Update  $w$  and  $b$  iteratively

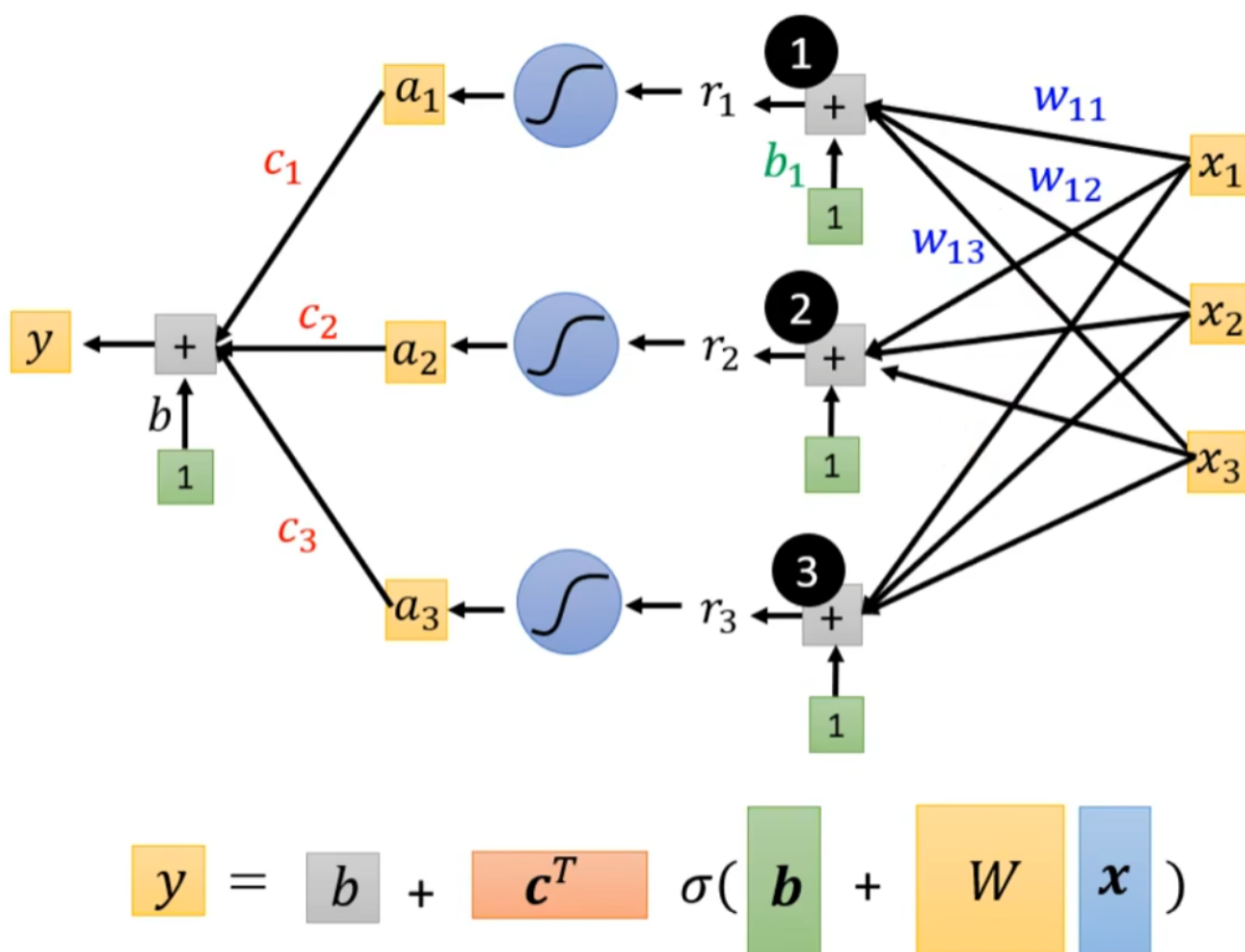
当移动为0（不能再偏移了）停下来 -> 找到了局部最优。问题是：可能找不到全局最好

(2) 经典线性模型太多简单，Model Bias -> 激活函数给与更多的非线性弹性特征

red curve = sum of a set of  + constant



计算展开可视化并转换成矩阵乘法：



在机器学习中，将所有未知向量**平展并拼成** $\theta$ ，那么损失函数就变成了计算 $L(\theta)$ ，此时 Optimization阶段公式可以写成：

# Optimization of New Model

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} L$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

➤ (Randomly) Pick initial values  $\boldsymbol{\theta}^0$

$$\text{gradient } \mathbf{g} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \frac{\partial L}{\partial \theta_2} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \theta_1^1 \\ \theta_2^1 \\ \vdots \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ \vdots \end{bmatrix} - \begin{bmatrix} \eta \frac{\partial L}{\partial \theta_1} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \eta \frac{\partial L}{\partial \theta_2} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

$$\mathbf{g} = \nabla L(\boldsymbol{\theta}^0)$$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \mathbf{g}$$

参数太多，偏移通常不会自动停下来。同时对于数据的训练，也会划分多个batch。

# Optimization of New Model

$$\theta^* = \arg \min_{\theta} L$$

➤ (Randomly) Pick initial values  $\theta^0$

➤ Compute gradient  $\mathbf{g} = \nabla L^1(\theta^0)$

$$\theta^1 \leftarrow \theta^0 - \eta \mathbf{g}$$

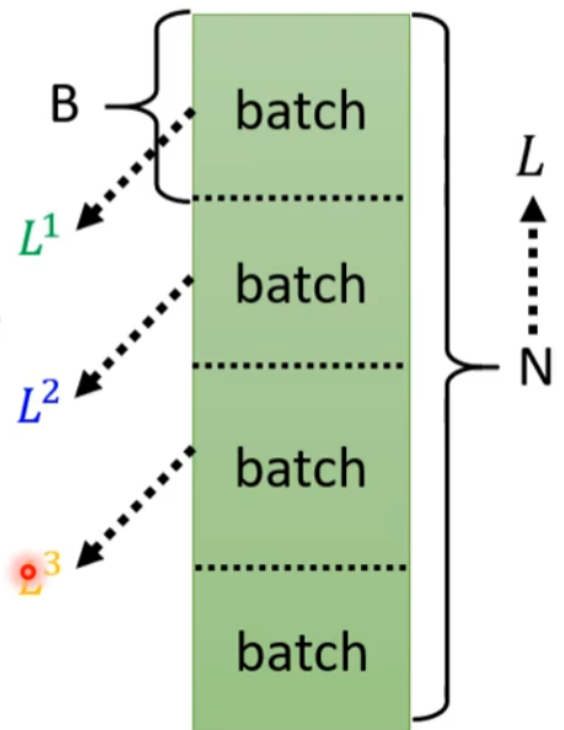
➤ Compute gradient  $\mathbf{g} = \nabla L^2(\theta^1)$

$$\theta^2 \leftarrow \theta^1 - \eta \mathbf{g}$$

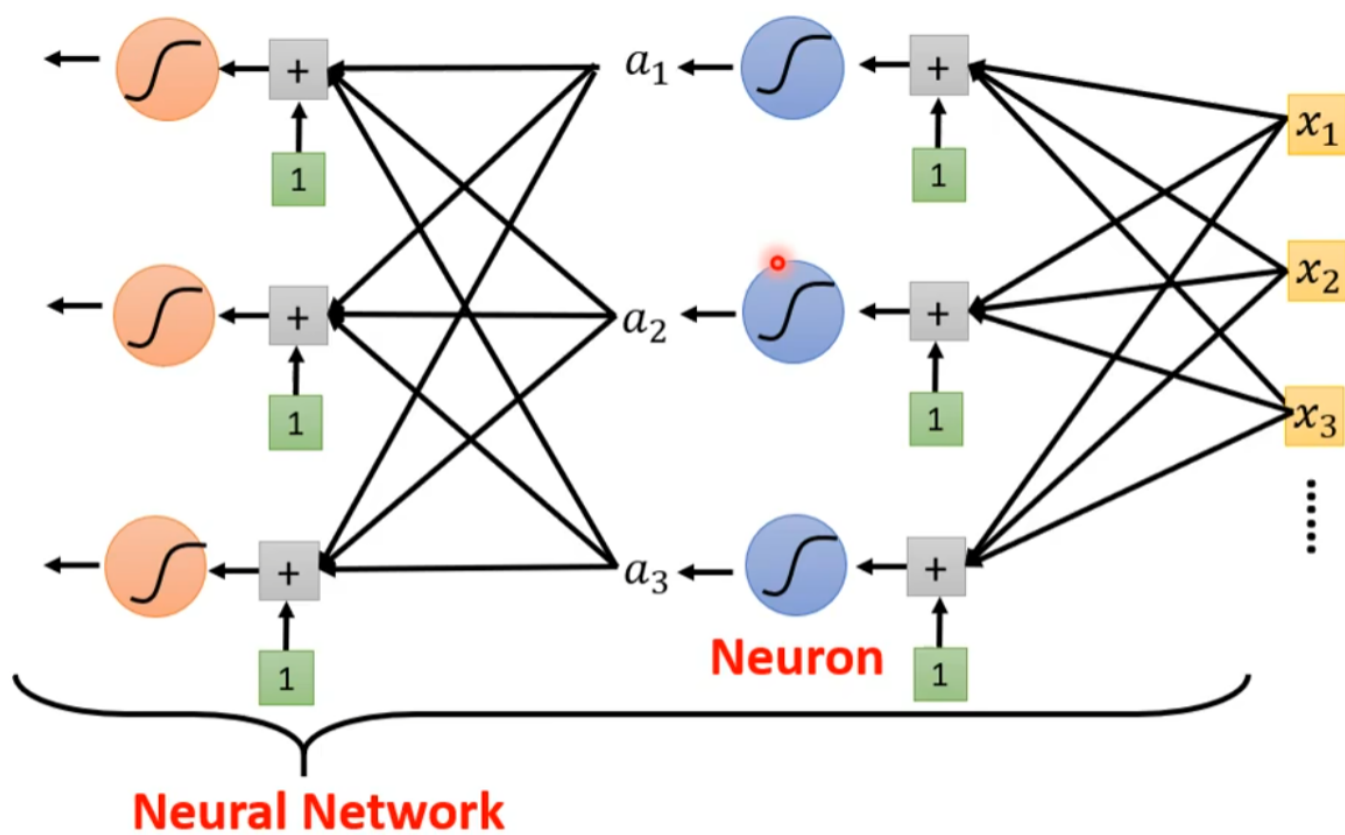
➤ Compute gradient  $\mathbf{g} = \nabla L^3(\theta^2)$

$$\theta^3 \leftarrow \theta^2 - \eta \mathbf{g}$$

**1 epoch** = see all the batches once



对于参数的计算还可以多做几次，就变成神经网络：



也会出现Overfitting：数据集表现好，训练集表现不好