



智能计算系统基础

2.神经网络基础理论

田晗 (henrytian@ustc.edu.cn) 特任副研究员

中国科学技术大学 计算机科学与技术学院

2024年9月



微信群：发布分组文档，上课信息等。



群聊：智能计算系统基础



该二维码 7 天内 (9 月 9 日前) 有效，重新进入将更新

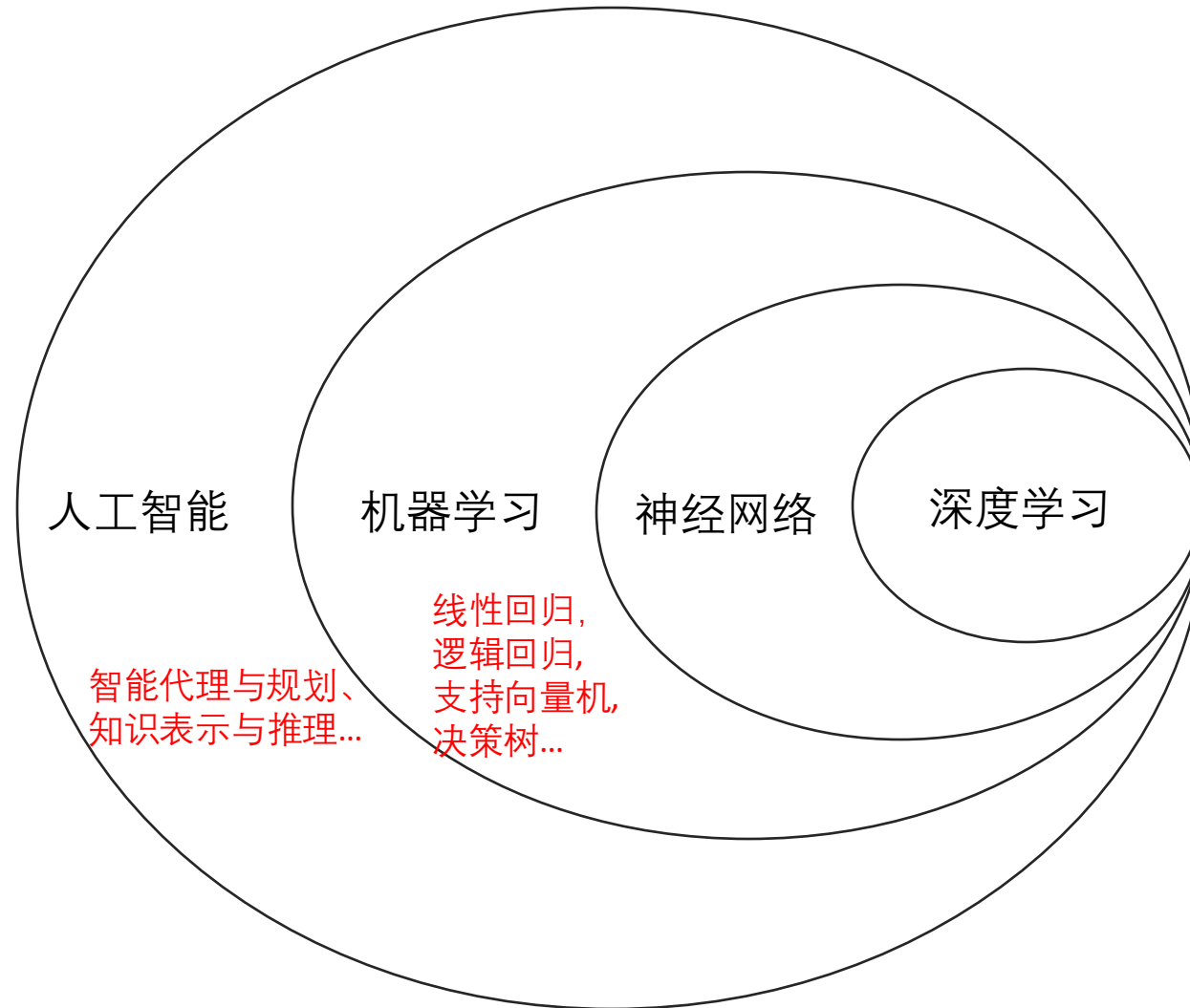


提纲

- 机器学习
- 神经网络
- 神经网络训练方法
- 神经网络设计基础
- 过拟合与正则化
- 交叉验证
- 本章小结

包含关系

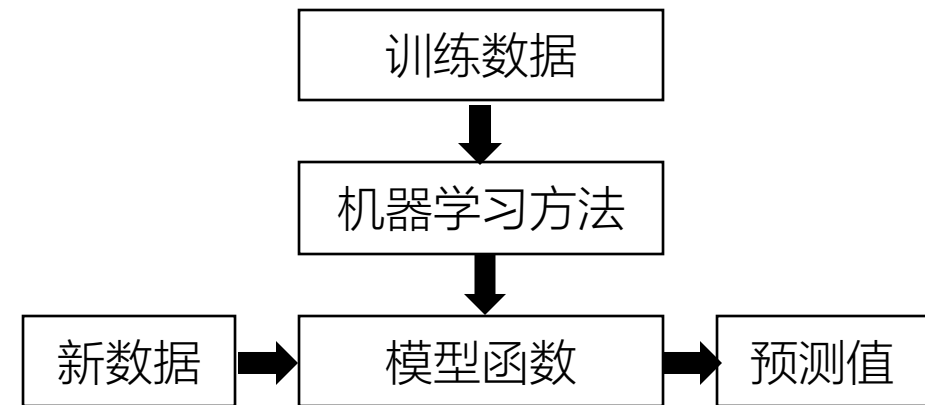
- 人工智能
- 机器学习
- 神经网络
- 深度学习



机器学习

- 机器学习：通过数据与机器学习方法，对数据内在规律（假设）进行优化与验证，从而获得对数据/任务模式最为准确的函数建模。
- 例子：认识天鹅

典型机器学习过程



从线性回归开始

问题引入：假设房屋销售中心有这样一组关于房屋面积和房屋位置与销售价格的数据，用 x_1 表示房屋面积， x_2 表示房屋楼层， y 表示售价（万元）。

x_1	50	47	60	55	...	65
x_2	2	1	4	3	...	10
...					...	
y	50	42	80	52	...	?

设计一个回归
程序进行预测

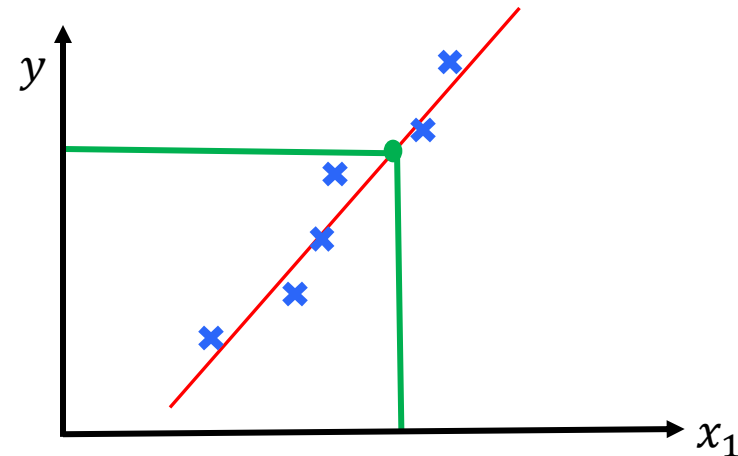
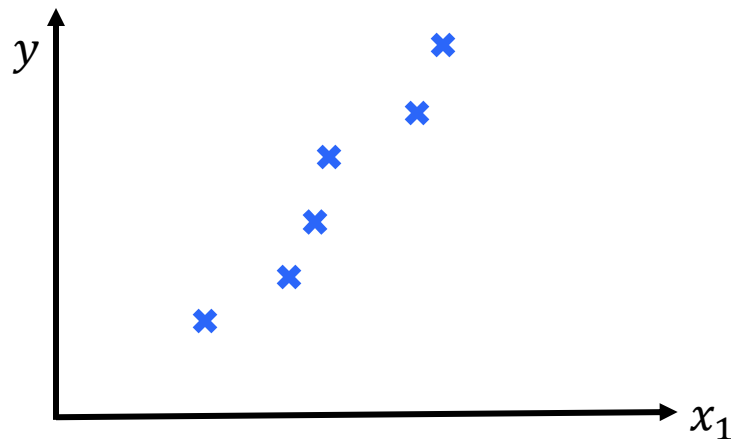
房屋面积 x_1 ，与售价 y 的关系

x_1	50	47	60	55	...	65
y	50	42	80	52	...	?

如果给出一个新的房屋面积，在销售记录中没有的，如何确定在该面积下的房屋售价 y 的值呢？

如预测 $x_1 = 65$ 时的售价？

解决办法->寻找 x_1 和 y 的之间的关系，使用已有记录数据进行训练



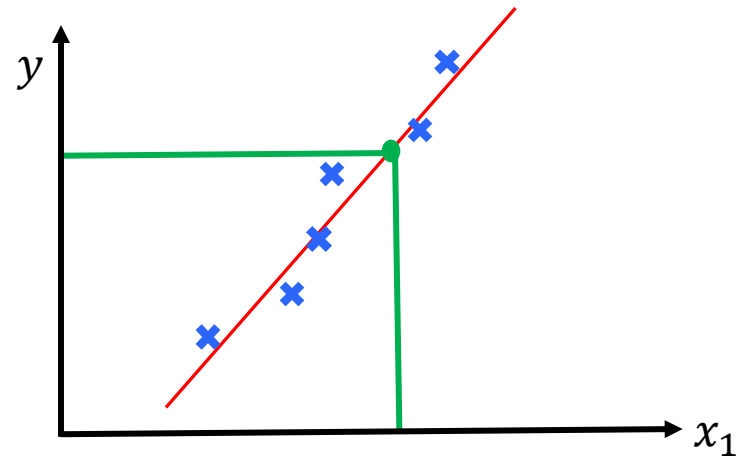
单变量线性回归模型（一元回归模型）

- 线性回归可以找到一些点的集合背后的规律：一个点集可以用一条直线来拟合，这条拟合出来的直线的参数特征，就是线性回归找到的点集背后的规律。

单变量线性模型

$$H_w(x) = w_0 + wx$$

x : Feature; $H(x)$: hypothesis



多变量线性回归模型

假设影响售价 y 的特征不仅仅只有房屋面积 x_1 ，还有楼层 x_2 ，房屋朝向 x_3 ， x_4 ，...， x_n

多变量线性模型

$$H_w(x) = w_0 + w_1x_1 + w_2x_2 \quad \text{2个特征}$$

单变量线性模型

$$H_w(x) = w_0 + wx$$



$$H_w(x) = \sum_{i=0}^n w_i x_i = \hat{\mathbf{w}}^T \mathbf{x}, \quad \text{n个特征}$$

$\hat{\mathbf{w}} = [w_0; w_1; \dots w_n]$ ，不同的权重意味着不同的模型假设

$$\mathbf{x} = [x_0; x_1; \dots x_n], \quad x_0 = 1$$



线性函数拟合好不好？

模型预测值 \hat{y} 与真实值 y 之间存在误差 $\varepsilon = y - \hat{y} = y - \hat{\mathbf{w}}^T \mathbf{x}$

我们可以通过修正模型权重向量 $\hat{\mathbf{w}}$ ，使得该误差尽可能的减少。

损失函数
$$L(\hat{\mathbf{w}}) = \frac{1}{2} \sum_{j=1}^m (H_{\mathbf{w}}(\mathbf{x}_j) - \mathbf{y}_j)^2 = \frac{1}{2} \sum_{j=1}^m (\hat{\mathbf{w}}^T \mathbf{x}_j - \mathbf{y}_j)^2$$

目标: 求出参数 $\hat{\mathbf{w}}$ ，使得损失函数 $L(\hat{\mathbf{w}})$ 取值最小

方法：解析解，梯度最速下降法，牛顿法...

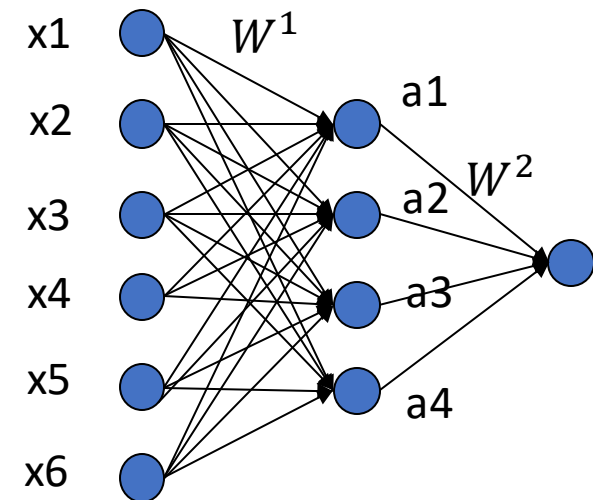
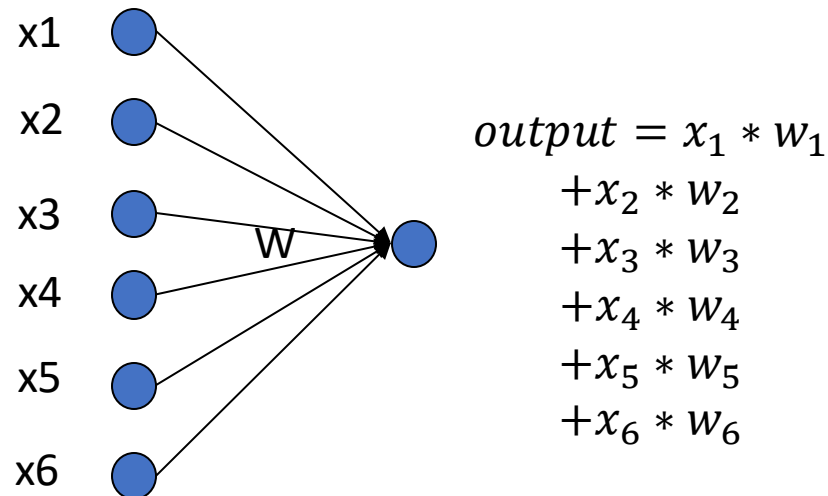


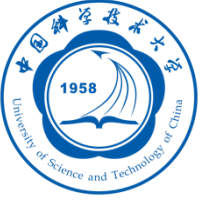
线性假设的偏差

- 模型假设：特征 x 与标签 y 之间的线性关系
- 当数据过于复杂时，将会引起明显偏差。

从线性回归到神经网络

- 线性回归模型的模型假设（或称为模型先验知识）过于简单，导致其无法很好的建模复杂问题（高偏差）。
- 因此，研究者基于权重加和这个简单操作，不断叠加，构建能够近似复杂函数的神经网络。



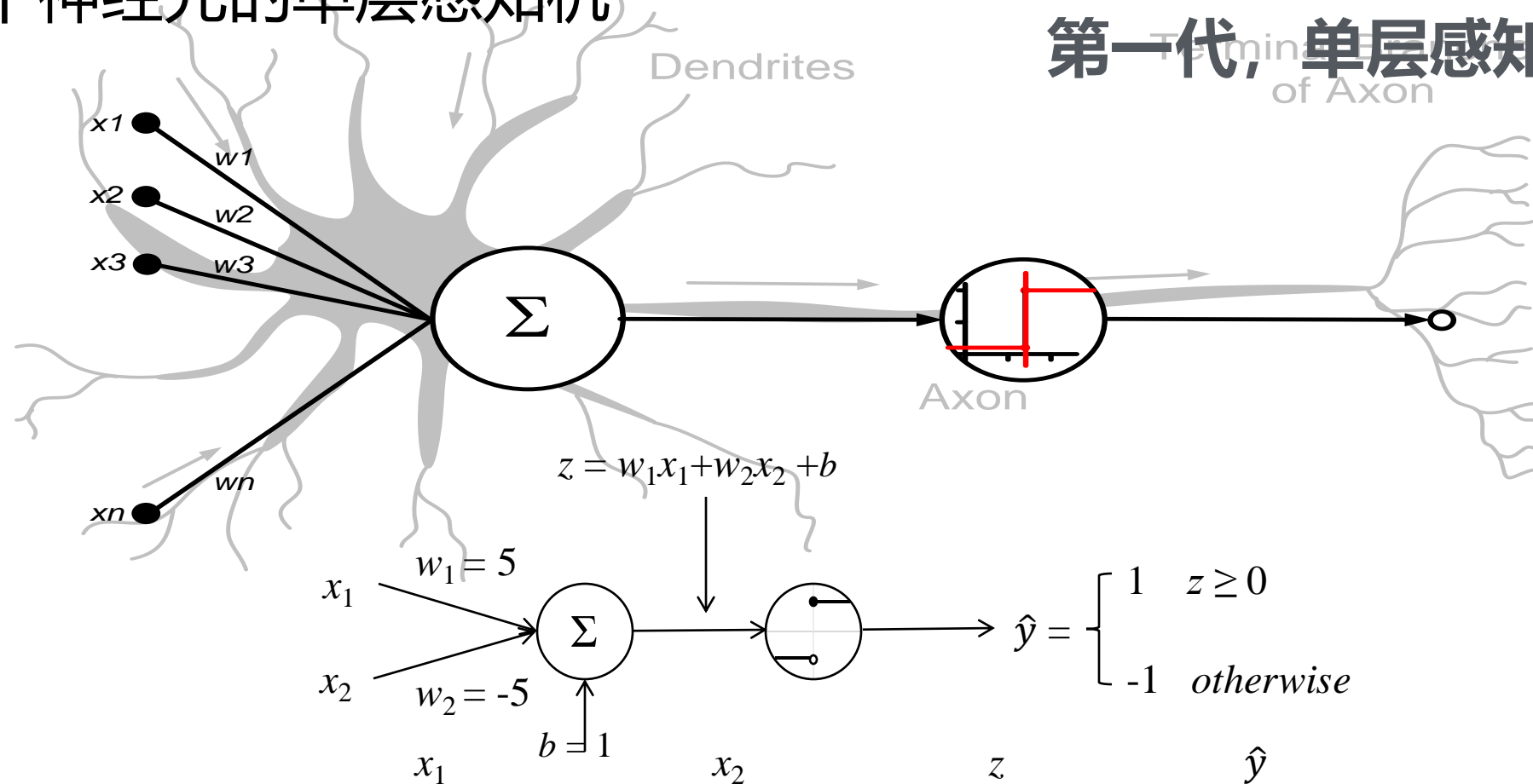






提纲

- 机器学习
- 神经网络
- 神经网络训练方法
- 神经网络设计基础
- 过拟合与正则化
- 交叉验证
- 本章小结

一个神经元的单层感知机

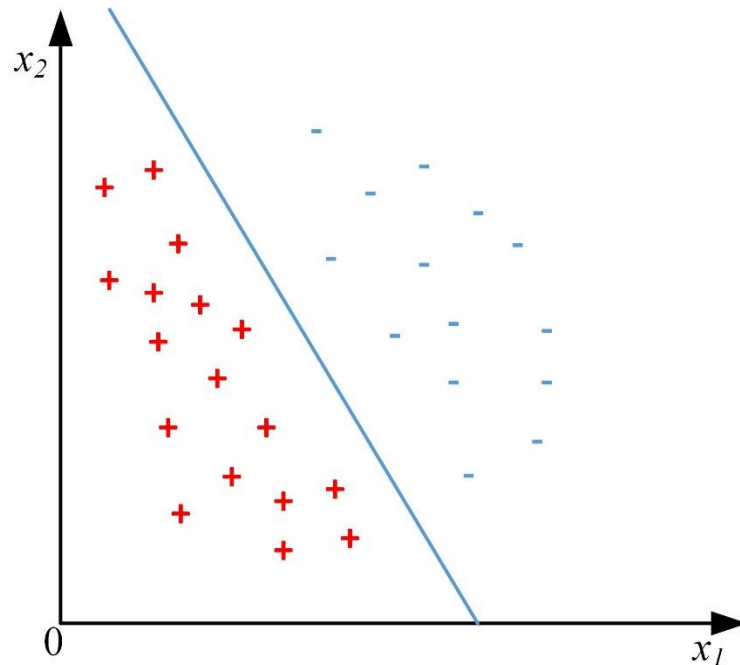
第一代, 单层感知机



	1		-1	11	1
	-1		1	-9	-1

感知机 (Perceptron) 模型

感知机模型 $H(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$ 对应一个超平面 $\mathbf{w}^T \mathbf{x} + b = 0$ ，模型参数是 (\mathbf{w}, b) 。感知机的目标是找到一个 (\mathbf{w}, b) ，将线性可分的数据集 T 中的所有样本点正确地分为两类。



$$H(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

$$\text{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$



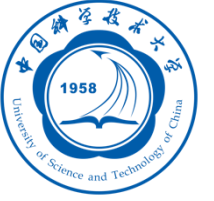
寻找损失函数，并
将损失函数最小化

➤ 分类问题的损失函数怎么做？

考虑一个训练数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ，其中， $\mathbf{x}_j \in \mathbb{R}^n$ ， $y_j \in \{+1, -1\}$ 。如果存在某个超平面 $S: (\mathbf{w}^T \mathbf{x} + b = 0)$ ，能将正负样本分到 S 两侧，则说明数据集可分，那么，如何求出这个超平面 S 的表达式？

策略：假设误分类的点为数据集 M ，使用误分类点到超平面的总距离来寻找损失函数（直观来看，总距离越小越好）

样本点 \mathbf{x}_j 到超平面 S 的距离：
$$d = \frac{1}{\|\mathbf{w}\|} |\mathbf{w}^T \mathbf{x}_j + b|$$
 $\|\mathbf{w}\|$ 是 \mathbf{w} 的 L^2 范数



➤ 寻找损失函数

数据集中误分类点满足条件： $-y_j(\mathbf{w}^T \mathbf{x}_j + b) > 0$

去掉点 \mathbf{x}_j 到超平面S的距离的绝对值符号：

$$d = -\frac{1}{\|\mathbf{w}\|} y_j(\mathbf{w}^T \mathbf{x}_j + b)$$

所有误分类点到超平面S的总距离为

$$d = -\frac{1}{\|\mathbf{w}\|} \sum_{\mathbf{x}_j \in M} y_j(\mathbf{w}^T \mathbf{x}_j + b)$$

由此寻找到感知机的损失函数 $L(\mathbf{w}, b) = - \sum_{\mathbf{x}_j \in M} y_j(\mathbf{w}^T \mathbf{x}_j + b)$



感知机算法

$$f(x) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

$$\text{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$



损失函数

$$L(\mathbf{w}, b) = - \sum_{\mathbf{x}_j \in M} y_j (\mathbf{w}^T \mathbf{x}_j + b)$$

问题转化为寻找 (\mathbf{w}, b) 使得损失函数极小化的最优化问题

对于训练集中的每一个样本 (x, y) :

计算当前分类器 $f(x) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$

如果 $f(x) \neq y$ (即分类错误), 则更新权重向量和偏置项:

$$\mathbf{w} = \mathbf{w} + \eta y \mathbf{x}$$

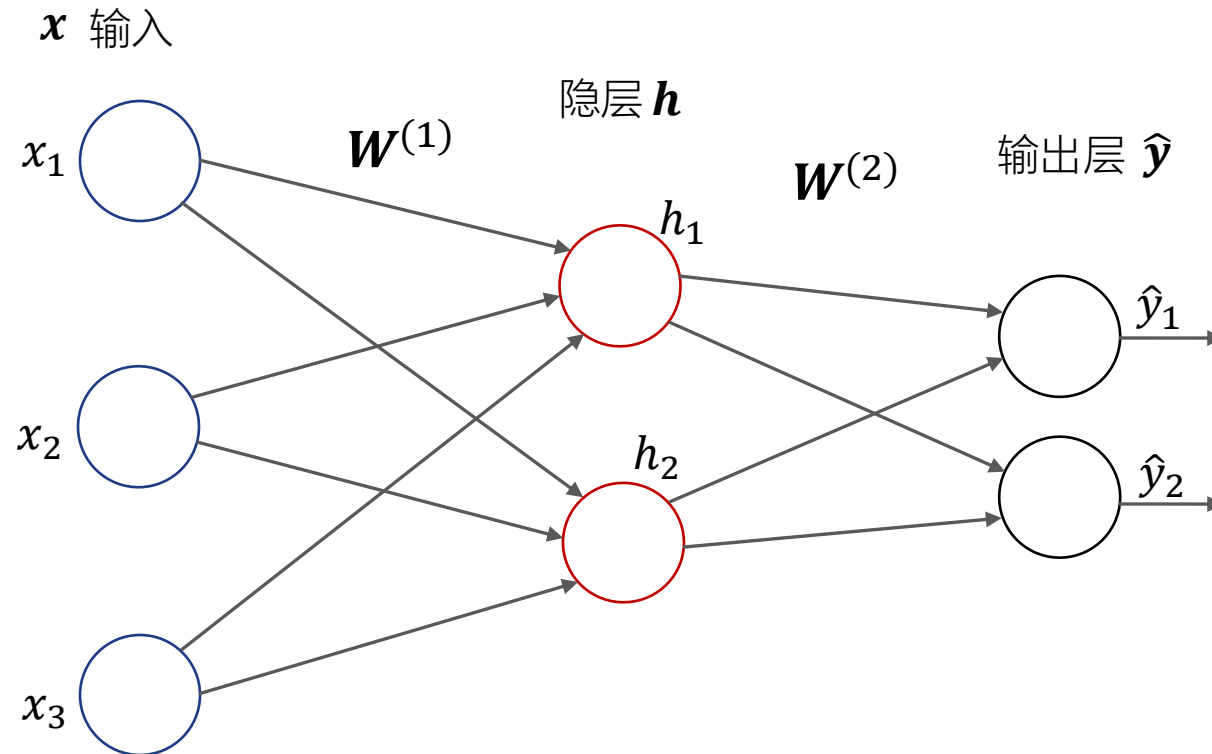
$$b = b + \eta y$$

其中 η 是学习率, 通常取较小的正值。

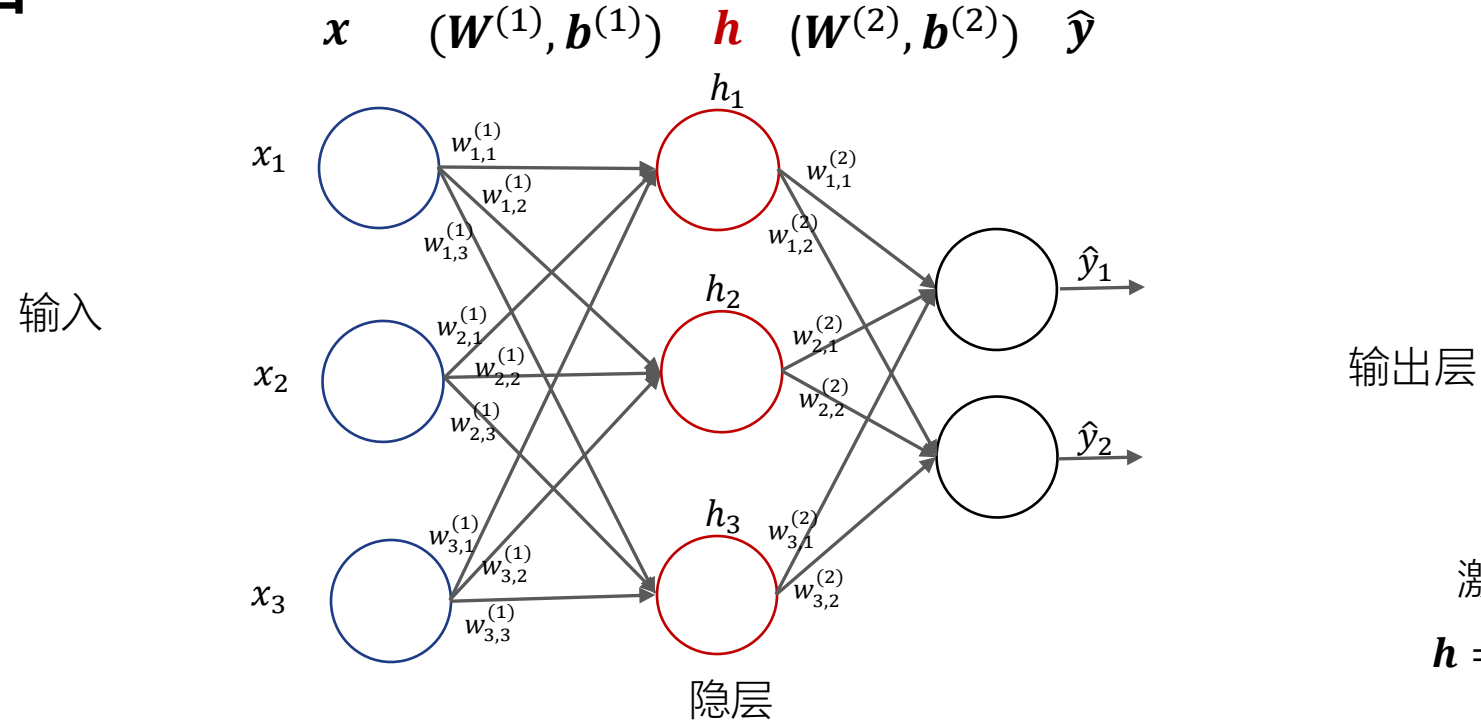
重复步骤 2, 直到所有训练样本都被正确分类或达到最大迭代次数。

两层神经网络-多层感知机

- 将大量的神经元模型进行组合，用不同的方法进行连接并作用在不同的激活函数上，就构成了人工神经网络模型。
- 全连接的两层神经网络模型也称为多层感知机（MLP）



正向传播



输入 $\mathbf{x} = [x_1; x_2; x_3]$

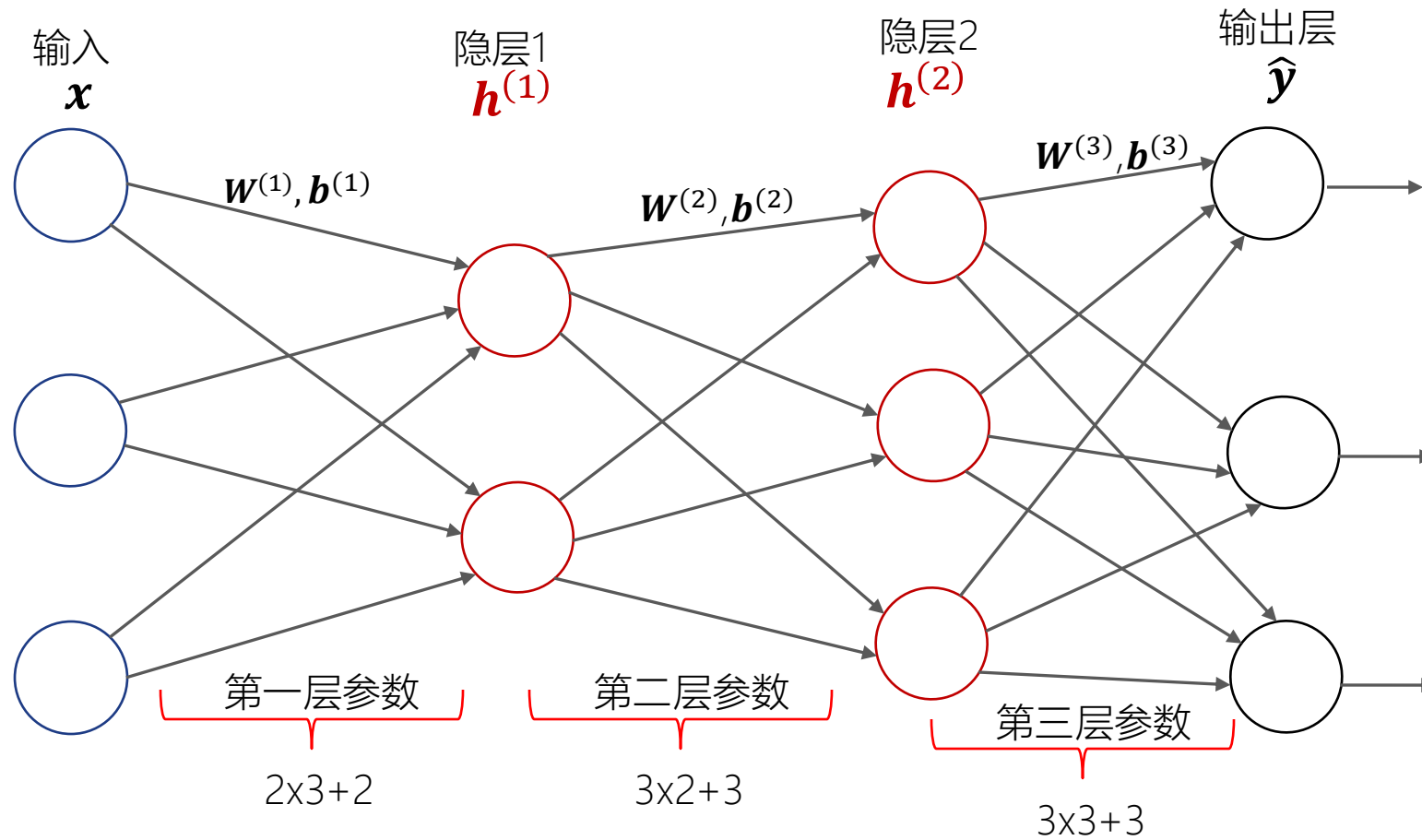
输入 $\mathbf{h} = [h_1; h_2; h_3]$

输出 $\hat{\mathbf{y}} = G(W^{(2)T} \mathbf{h} + \mathbf{b}^{(2)})$

$$\text{权重 } W^{(1)} = \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & w_{2,3}^{(1)} \\ w_{3,1}^{(1)} & w_{3,2}^{(1)} & w_{3,3}^{(1)} \end{bmatrix}$$

$$\text{权重 } W^{(2)} = \begin{bmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} \\ w_{3,1}^{(2)} & w_{3,2}^{(2)} \end{bmatrix}$$

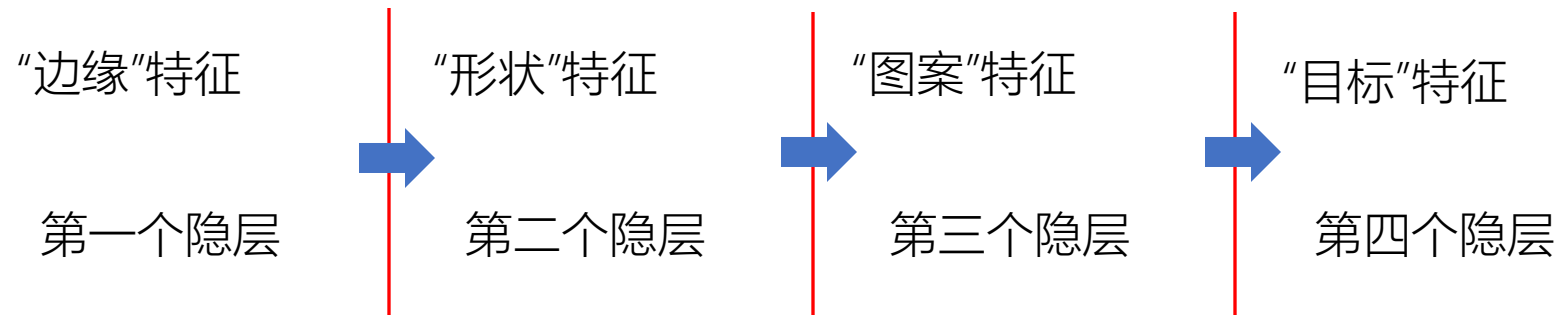
多层神经网络



需 $6+6+9+8=29$ 个参数

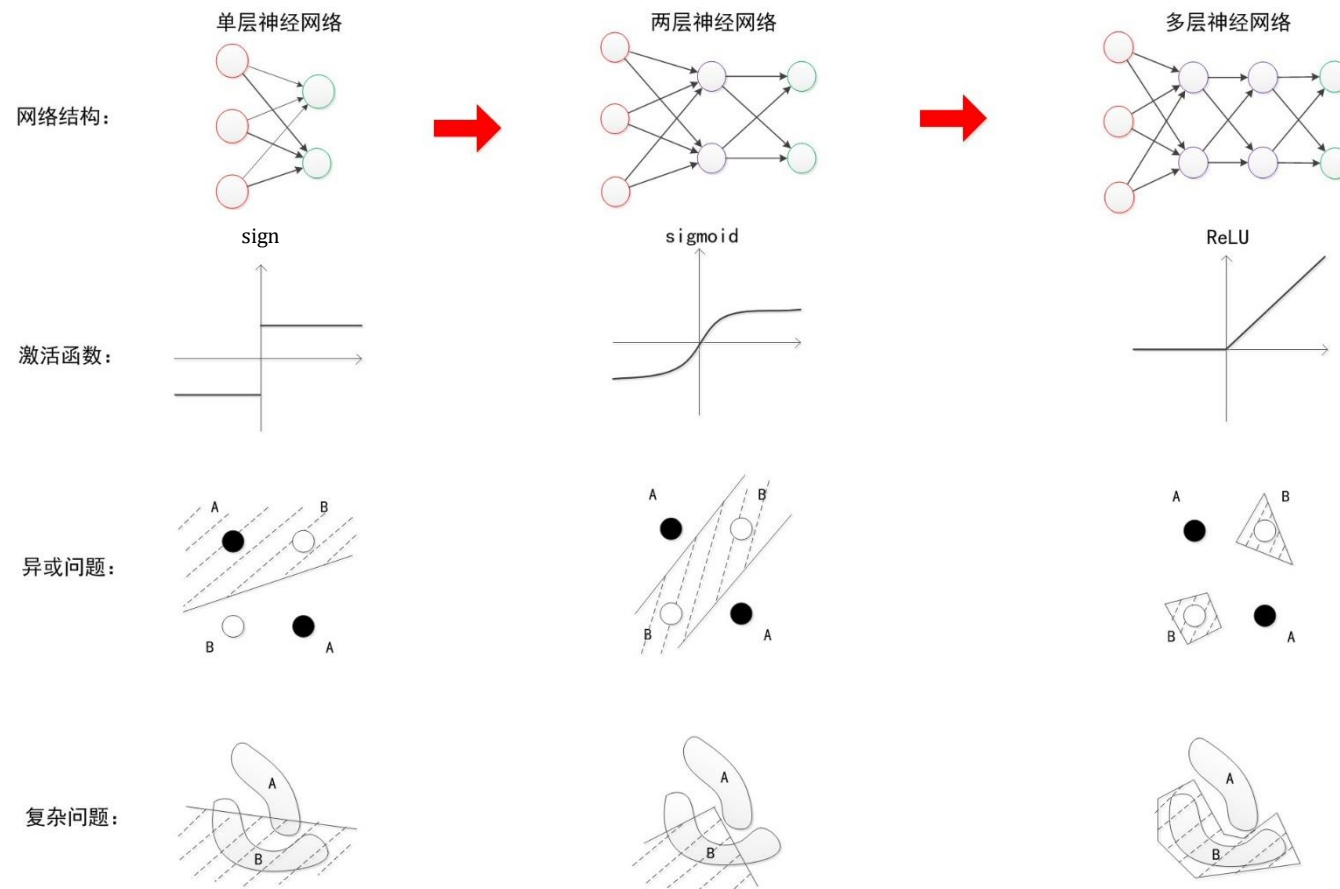
多层神经网络

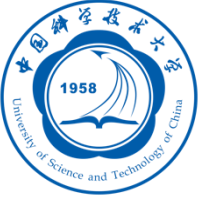
- 随着网络的层数增加，每一层对于前一层次的抽象表示更深入，每一层神经元学习到的是前一层神经元更抽象的表示
- 通过抽取更抽象的特征来对事物进行区分，从而获得更好的区分与分类能力



多层神经网络

从单层神经网络，到两层神经网络，再到多层神经网络，随着网络层数的增加，以及激活函数的调整，神经网络拟合非线性分界不断增强。





Why Go Deeper?

Kurt Hornik证明了理论上两层神经网络足以拟合任意函数

其局限性在于对复杂函数的表示“效率”有限（虽浅但宽），
因此**训练效率很低**。（似快实慢）

多层神经网络问题：训练难以收敛，梯度回传容易爆炸或者消失

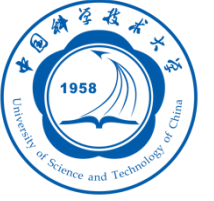
深度学习（深层神经网络）

2006年，Hinton在Science发表了论文（Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786），给多层神经网络相关的学习方法赋予了一个新名词--“深度学习”。他和LeCun以及Bengio三人被称为深度学习三位开创者



...

Geoffery Hinton

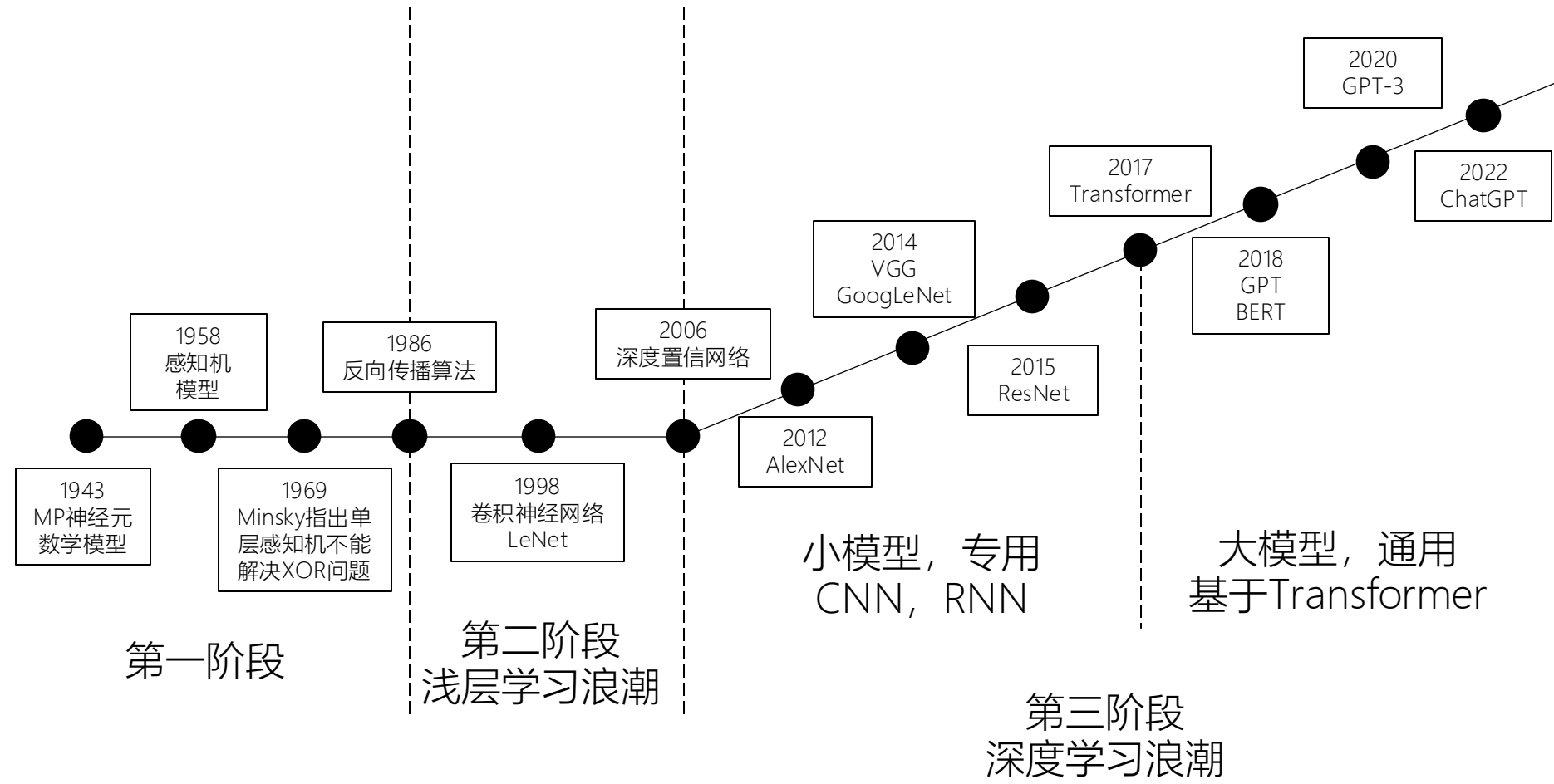


深度神经网络的成功：ABC

深度神经网络不断发展不仅依赖于自身的结构优势，也依赖于如下一些外在因素

- Algorithm: 算法日新月异，优化算法层出不穷（学习算法->BP 算法-> Pre-training, Dropout等方法)
- Big data: 数据量不断增大 (10-> 10 k ->100M)
- Computing: 处理器计算能力的不断提升（晶体管->CPU ->集群/GPU ->智能处理器)

深度学习发展历程

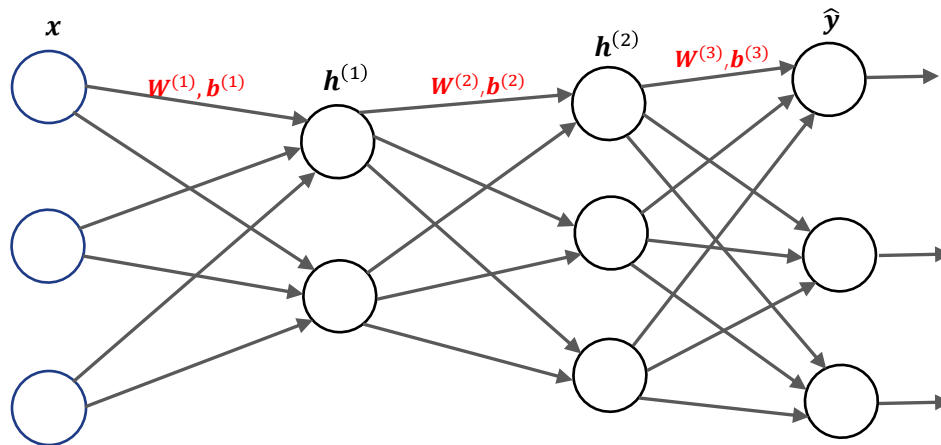




提纲

- 机器学习
- 神经网络
- 神经网络训练方法
- 神经网络设计基础
- 过拟合与正则化
- 交叉验证
- 本章小结

神经网络的模型训练

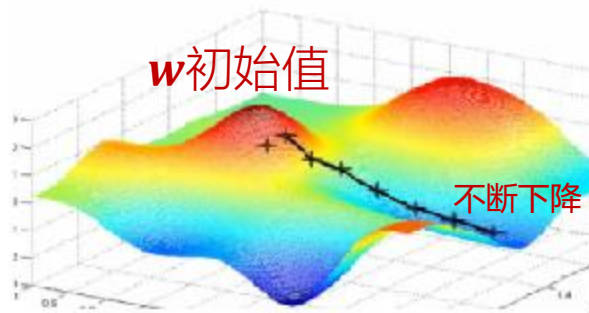


- 模型训练的目的，就是调整参数使得模型计算值 \hat{y} 尽可能的与真实值 y 逼近

寻找参数 \hat{w} ->使得 $L(\hat{w})$ 最小

迭代法（梯度下降法）寻找参数

- 初始先给定一个 \hat{w} ，如 $\mathbf{0}$ 向量或随机向量
- 沿着梯度下降的方向进行迭代，使更新后的 $L(\hat{w})$ 不断变小



$$\hat{w} = \hat{w} - \alpha \frac{\partial L(\hat{w})}{\partial \hat{w}}$$

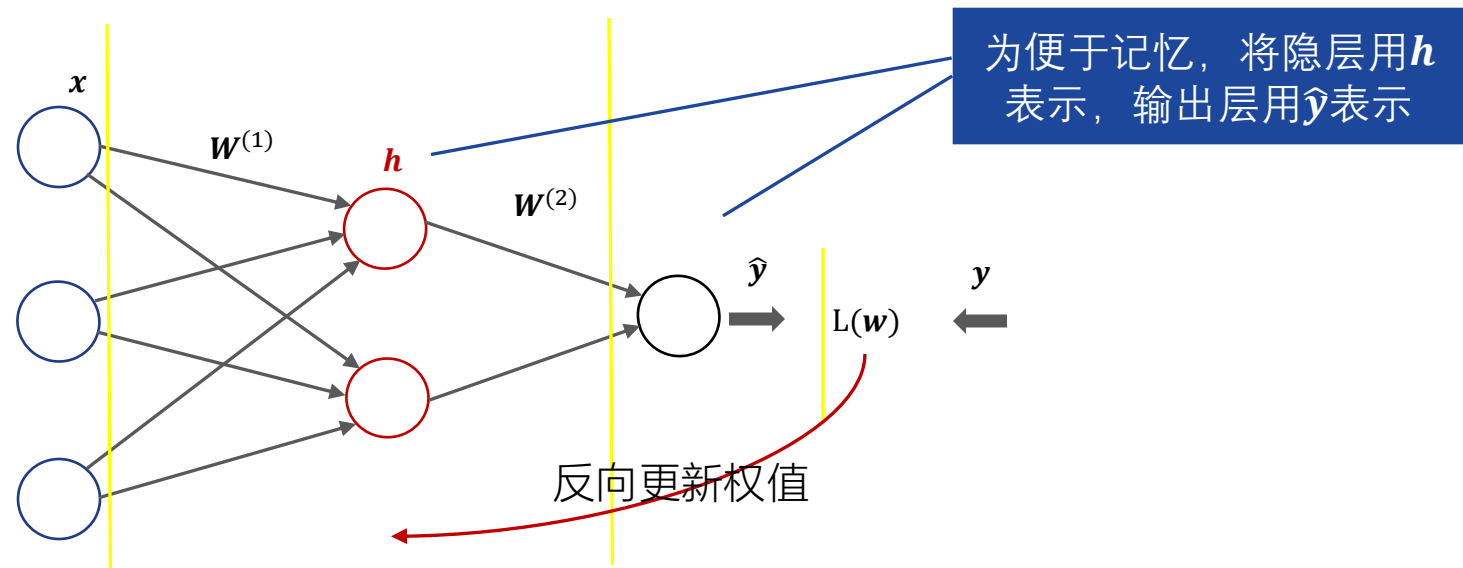
α 称为学习率或步长



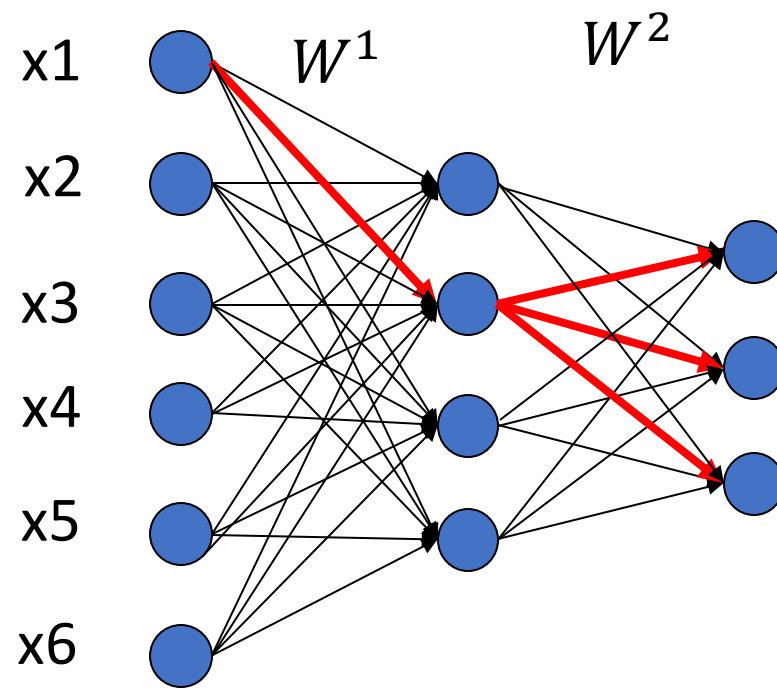
迭代至找到使得 $L(\hat{w})$ 最小的 \hat{w} 值停止，从而得到回归模型参数

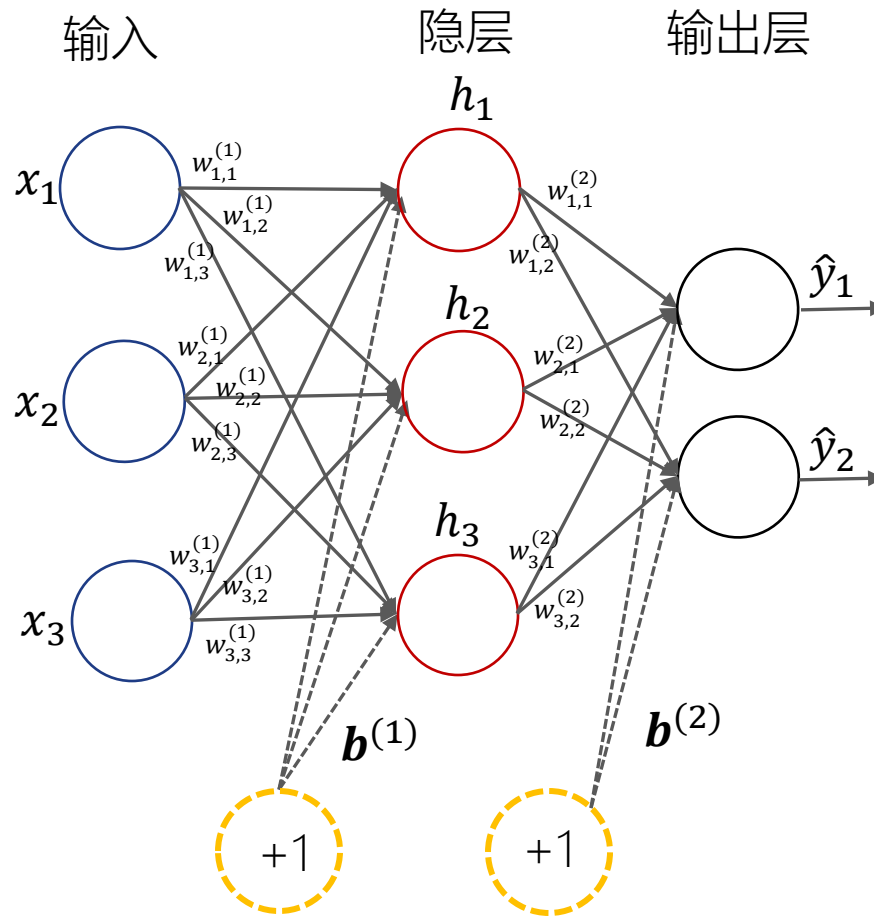
神经网络训练

- 正向传播（推断）是根据输入，经过权重、激活函数计算出隐层，将输入的特征向量从低级特征逐步提取为抽象特征，直到得到最终输出结果的过程。

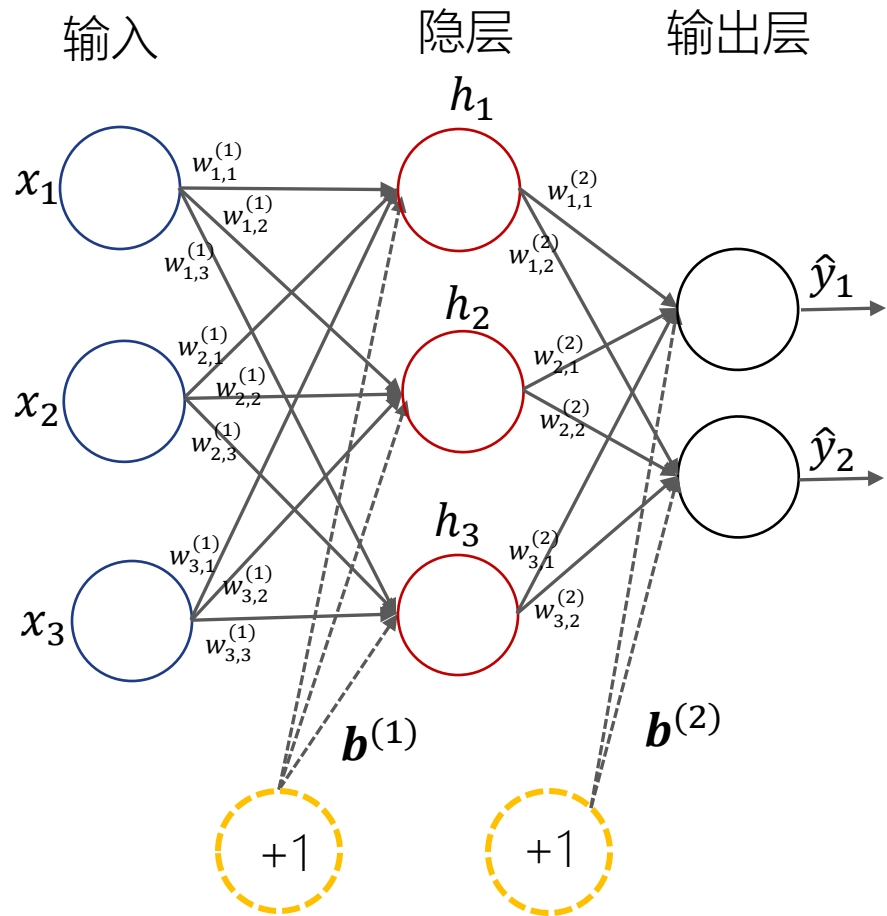


- 反向传播是根据正向传播的输出结果和期望值计算出损失函数，再通过链式求导，最终从网络后端逐步修改权重使输出和期望值的差距变到最小的过程。





- 输入：包含神经元 x_1, x_2, x_3
- 隐层：包含 h_1, h_2, h_3
- 输出层：包含 \hat{y}_1, \hat{y}_2
- 输入和隐层之间
偏置： $b^{(1)}$
权重： $W^{(1)}$
- 隐层和输出层之间
偏置： $b^{(2)}$
权重： $W^{(2)}$



- 使用 sigmoid 函数作为激活函数与最后的输出调整函数

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

前向传输：输入到隐层

$$\begin{aligned} \mathbf{v} &= \mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)} \\ &= \begin{bmatrix} w_{1,1}^{(1)} & w_{2,1}^{(1)} & w_{3,1}^{(1)} \\ w_{1,2}^{(1)} & w_{2,2}^{(1)} & w_{3,2}^{(1)} \\ w_{1,3}^{(1)} & w_{2,3}^{(1)} & w_{3,3}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \mathbf{b}^{(1)} \end{aligned}$$

$$\mathbf{h} = \frac{1}{1 + e^{-\mathbf{v}}}$$

示例

假定输入数据 $x_1 = 0.02$ 、 $x_2 = 0.04$ 、 $x_3 = 0.01$

固定偏置 $\mathbf{b}^{(1)} = [0.4; 0.4; 0.4]$ 、 $\mathbf{b}^{(2)} = [0.7; 0.7]$

期望输出 $y_1 = 0.9$ 、 $y_2 = 0.5$

未知权重

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & w_{2,3}^{(1)} \\ w_{3,1}^{(1)} & w_{3,2}^{(1)} & w_{3,3}^{(1)} \end{bmatrix}, \quad \mathbf{W}^{(2)} = \begin{bmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} \\ w_{3,1}^{(2)} & w_{3,2}^{(2)} \end{bmatrix}$$

目的是为了能得到 $y_1 = 0.9$ 、 $y_2 = 0.5$ 的期望的值，需计算出合适的 $\mathbf{W}^{(1)}$ 、 $\mathbf{W}^{(2)}$ 的权重值

➤ 初始化权重值

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & w_{2,3}^{(1)} \\ w_{3,1}^{(1)} & w_{3,2}^{(1)} & w_{3,3}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.25 & 0.15 & 0.30 \\ 0.25 & 0.20 & 0.35 \\ 0.10 & 0.25 & 0.15 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} \\ w_{3,1}^{(2)} & w_{3,2}^{(2)} \end{bmatrix} = \begin{bmatrix} 0.40 & 0.25 \\ 0.35 & 0.30 \\ 0.01 & 0.35 \end{bmatrix}$$

➤ 输入到隐层计算

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)} = \begin{bmatrix} 0.25 & 0.25 & 0.10 \\ 0.15 & 0.20 & 0.25 \\ 0.30 & 0.35 & 0.15 \end{bmatrix} \begin{bmatrix} 0.02 \\ 0.04 \\ 0.01 \end{bmatrix} + \begin{bmatrix} 0.4 \\ 0.4 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.4160 \\ 0.4135 \\ 0.4215 \end{bmatrix}$$

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \frac{1}{1 + e^{-v}} = \begin{bmatrix} \frac{1}{1 + e^{-0.416}} \\ \frac{1}{1 + e^{-0.4135}} \\ \frac{1}{1 + e^{-0.4215}} \end{bmatrix} = \begin{bmatrix} 0.6025 \\ 0.6019 \\ 0.6038 \end{bmatrix}$$

➤ 隐层到输出层计算

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \mathbf{W}^{(2)T} \mathbf{h} + \mathbf{b}^{(2)} = \begin{bmatrix} 0.40 & 0.35 & 0.01 \\ 0.25 & 0.30 & 0.35 \end{bmatrix} \begin{bmatrix} 0.6025 \\ 0.6019 \\ 0.6038 \end{bmatrix} + \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} = \begin{bmatrix} 1.1577 \\ 1.2425 \end{bmatrix}$$

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \frac{1}{1 + e^{-z}} = \begin{bmatrix} \frac{1}{1 + e^{-1.1577}} \\ \frac{1}{1 + e^{-1.2425}} \end{bmatrix} = \begin{bmatrix} 0.7609 \\ 0.7760 \end{bmatrix}$$

↑
距离期望输出 $y_1 = 0.9$ 、
 $y_2 = 0.5$ 还有差距，通过反向传播修改权重

模型计算输出

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} 0.7609 \\ 0.7760 \end{bmatrix}$$

期望输出

$$y_1 = 0.9$$

$$y_2 = 0.5$$

➤ 计算误差

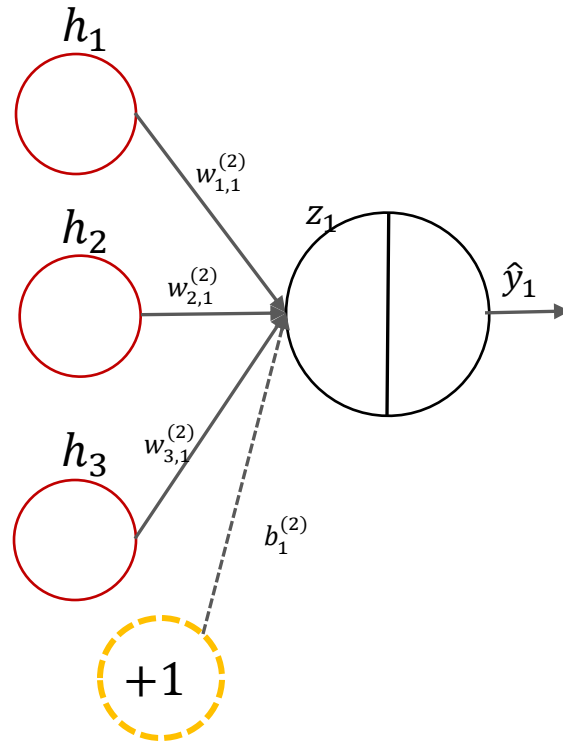
$$\begin{aligned} L(W) &= L_1 + L_2 = \frac{1}{2} (y_1 - \hat{y}_1)^2 + \frac{1}{2} (y_2 - \hat{y}_2)^2 \\ &= \frac{1}{2} (0.7609 - 0.9)^2 + \frac{1}{2} (0.7760 - 0.5)^2 = 0.0478 \end{aligned}$$

计算值与真实值之间还有很大的差距，如何缩小计算值与真实值之间的误差？

通过反向传播进行反馈，调节权重值

反向传播

- 隐层到输出层的权重 $\mathbf{W}^{(2)}$ 的更新



$$L(\mathbf{W}) = L_1 + L_2 = \frac{1}{2} (y_1 - \hat{y}_1)^2 + \frac{1}{2} (y_2 - \hat{y}_2)^2$$

- 以 $w_{2,1}^{(2)}$ (记为 ω) 参数为例子, 计算 ω 对整体误差的影响有多大, 可以使用整体误差对 ω 参数求偏导

➤ 根据偏导数的链式法则推导

$$\frac{\partial L(\mathbf{W})}{\partial \omega} = \frac{\partial L(\mathbf{W})}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial z_1} \frac{\partial z_1}{\partial \omega}$$

$$L(\mathbf{W}) = \frac{1}{2} (y_1 - \hat{y}_1)^2 + \frac{1}{2} (y_2 - \hat{y}_2)^2$$

$$\frac{\partial L(\mathbf{W})}{\partial \hat{y}_1} = -(y_1 - \hat{y}_1) = -(0.9 - 0.7609) = -0.1391$$

$$\hat{y}_1 = \frac{1}{1 + e^{-z_1}}$$

$$\frac{\partial \hat{y}_1}{\partial z_1} = \hat{y}_1(1 - \hat{y}_1) = 0.7609 * (1 - 0.7609) = 0.1819$$

➤ 根据偏导数的链式法则推导

$$\frac{\partial L(\mathbf{W})}{\partial \omega} = \frac{\partial L(\mathbf{W})}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial z_1} \frac{\partial z_1}{\partial \omega}$$

$$z_1 = w_{1,1}^{(2)} \times h_1 + \omega \times h_2 + w_{3,1}^{(2)} \times h_3 + b_1^{(2)}$$

$$\frac{\partial z_1}{\partial \omega} = h_2 = 0.6019$$



$$\begin{aligned} \frac{\partial L(\mathbf{W})}{\partial \omega} &= -(y_1 - \hat{y}_1) \times \hat{y}_1 (1 - \hat{y}_1) \times h_2 \\ &= -0.1391 \times 0.1819 \times 0.6019 = -0.0152 \end{aligned}$$

➤ 更新 $w_{2,1}^{(2)}$ (记为 ω) 的值

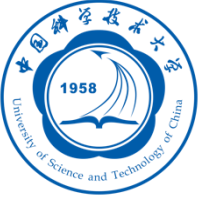
$$\mathbf{W}^{(2)} = \begin{bmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} \\ \color{red}{w_{2,1}^{(2)}} & w_{2,2}^{(2)} \\ w_{3,1}^{(2)} & w_{3,2}^{(2)} \end{bmatrix} = \begin{bmatrix} 0.40 & 0.25 \\ \color{red}{0.35} & 0.30 \\ 0.01 & 0.35 \end{bmatrix}$$

初始值

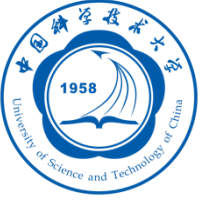
$$\frac{\partial L(\mathbf{W})}{\partial \omega} = -0.0152$$

$$\omega = \omega - \alpha \times \frac{\partial L(\mathbf{W})}{\partial \omega} = 0.35 - (-0.0152) = 0.3652$$

➤ 同理，可以计算新的 $\mathbf{W}^{(2)}$ 的其他元素的权重值



- 反向传播的作用是将神经网络的输出误差反向传播到神经网络的输入端，并以此来更新神经网络中各个连接的权重
- 当第一次反向传播法完成后，网络的模型参数得到更新，网络进行下一轮的正向传播过程，如此反复的迭代进行训练，从而不断缩小计算值与真实值之间的误差。



提纲

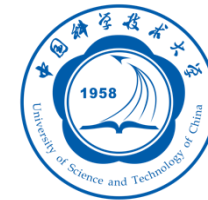
- 机器学习
- 神经网络
- 神经网络训练方法
- 神经网络设计基础
- 过拟合与正则化
- 交叉验证
- 本章小结



神经网络的模型训练

训练完了结果就是不准，怎么办？

- 调整网络拓扑结构
- 选择合适的激活函数
- 选择合适的损失函数



神经网络的拓扑调节

神经网络的结构一般为：输入×隐层×输出层

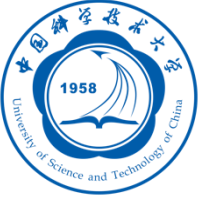
输入：神经元个数=特征维度

输出层：神经元个数=分类类别数

隐层：

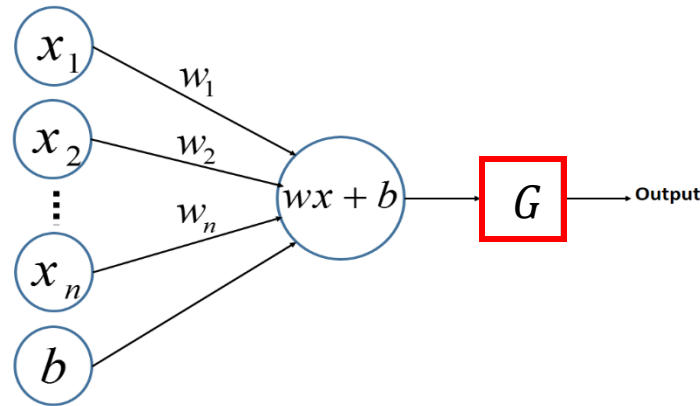
- 隐层的数量？
- 隐层神经元的个数？

给定训练样本后，输入和输出层节点数便已确定



- 隐层的设计:
 - 隐层节点的作用是提取输入特征中的隐藏规律，每个节点都赋予一定权重
 - 隐层节点数太少，则网络从样本中获取信息的能力就越差，无法反映数据集的规律；隐层节点数太多，则网络的拟合能力过强，可能拟合数据集中的噪声部分，导致模型泛化能力变差。

选择合适的激活函数



- 在神经元中，输入的数据通过加权求和后，还被作用了一个函数 G ，这个函数 G 就是激活函数（Activation Function）。
- 激活函数给神经元引入了非线性因素，使得神经网络可以任意逼近任何非线性函数，因此神经网络可以应用到众多的非线性模型中
- 激活函数需具备的性质
 - **可微性**：当优化方法是基于梯度的时候，这个性质是必须的。
 - **输出值的范围**：当激活函数输出值是有限的时候，基于梯度的优化方法会更加稳定，因为特征的表示受有限权值的影响更显著；当激活函数的输出是无限的时候，模型的训练会更加高效，不过在这种情况下，一般需要更小的学习率。

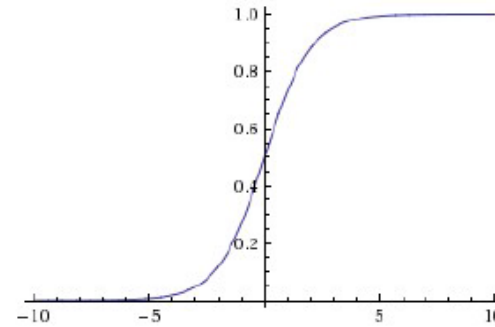
sigmoid函数

数学表达式

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Sigmoid 是最常见的非线性激活函数
- 能够把输入的连续实值变换为0和1之间的输出；如果是非常大的负数，那么输出就变为0；如果是非常大的正数，输出就变为1。

几何图像

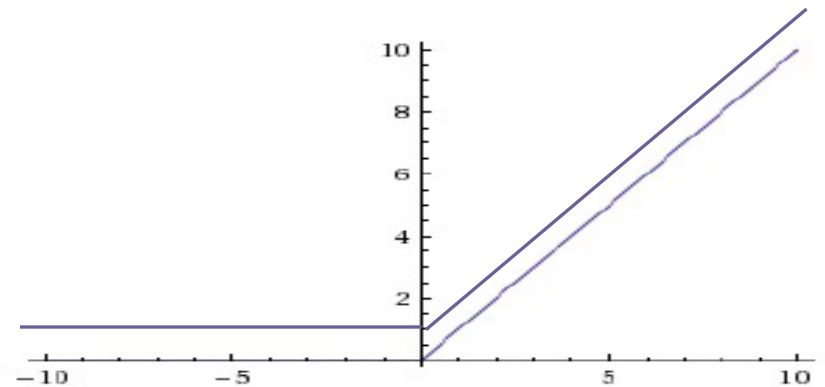


- 计算机进行指数运算速度慢
- 饱和性问题及梯度消失现象

ReLU函数

- ReLU 能够在 $x > 0$ 时保持梯度不衰减，从而缓解梯度消失问题
- 输出范围是无限的
- 如果学习率很大，反向传播后的参数可能为负数，导致下一轮正向传播的输入为负数。当输入是负数的时候，ReLU是完全不被激活的

$$f(x) = \max(0, x)$$



PReLU/Leaky ReLU 函数

ReLU 在 $x < 0$ 时,
ReLU 完全不被激活

改进

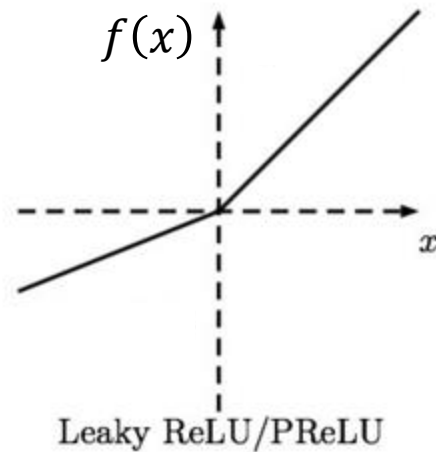


出现了ReLU的改进版本:

Leaky ReLU

$$f(x) = \max(\alpha x, x), \quad \alpha \in (0, 1)$$

- 负数区域内, Leaky ReLU有一个很小的斜率, 可以避免ReLU死掉的问题

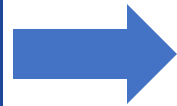


PReLU定义类似

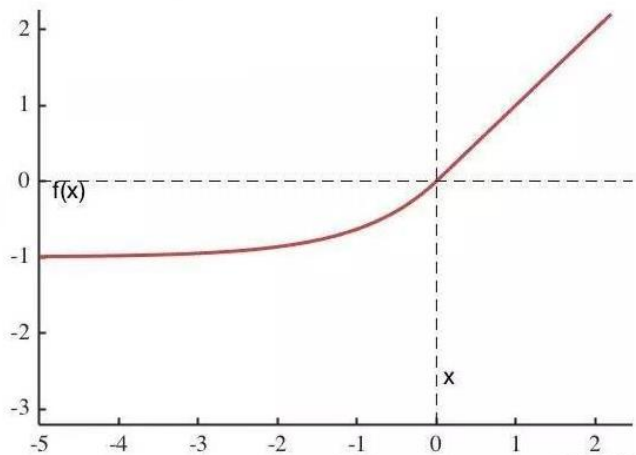
- α 为可调参数, 每个通道有一个 α , 反向传播训练得到

ELU函数(Exponential Linear Unit)

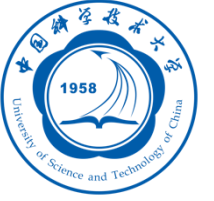
融合sigmoid和ReLU



$$\text{ELU: } f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$$



- α 是可调参数，控制着ELU在负值区间的饱和位置
- ELU的输出均值接近于零，所以收敛速度更快
- 右侧线性部分使得ELU能够缓解梯度消失，而左侧让ELU对输入变化或噪声更鲁棒，避免神经元死掉

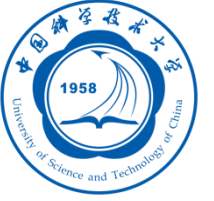


选择恰当的损失函数

损失函数 $L = f(\hat{y}, y)$, \hat{y} 是模型预测值, 是神经网络

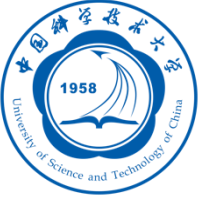
模型参数 W 的函数, 记作 $\hat{y} = H_w(x)$

从 \mathbf{w} 角度看, 损失函数可以记为 $L(\mathbf{w}) = f(H_w(x), y)$



神经网络中损失函数的特性

- 同一个算法的损失函数不是唯一的
- 损失函数是参数(w, b)的函数
- 损失函数可以评价网络模型的好坏，损失函数越小说明模型和参数越符合训练样本(x, y)
- 损失函数是一个标量
- 选择损失函数时，挑选对参数(w, b)可微的函数（全微分存在，偏导数一定存在）
- 损失函数又称为代价函数、目标函数



代价函数的选择：回归

Square loss: $L(\mathbf{w}, b) = \sum_{x_j \in D} (\hat{y}_j - y_j)^2$

输出 \hat{y}_j 力图尽可能的输出 y_j 。

分类问题如何使用随机梯度下降法?

$$L(\mathbf{w}, b) = - \sum_{x_j \in M} y_j (\mathbf{w}^T \mathbf{x}_j + b)$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b) = - \sum_{x_j \in M} y_j \mathbf{x}_j \quad \rightarrow \quad \mathbf{w} \leftarrow \mathbf{w} + \alpha y_j \mathbf{x}_j$$

$$\nabla_b L(\mathbf{w}, b) = - \sum_{x_j \in M} y_j \quad \rightarrow \quad b \leftarrow b + \alpha y_j$$

随机选取误分类点 (x_j, y_j)
对 \mathbf{w}, b 以 α 为步长进行更新, 通过迭代可以使得
损失函数 $L(\mathbf{w}, b)$ 不断减小, 直到为0

$$L(\mathbf{w}, b) \rightarrow 0$$

代价函数的选择：回归

梯度下降法需要对 w 可导的 L ，要么我们挨个找出来误分类点 M ，要么必须面对一个不可导的函数：

$$L(\mathbf{w}, b) = - \sum_{\mathbf{x}_j \in D} y_j (\mathbf{w}^T \mathbf{x}_j + b) [-y_j (\mathbf{w}^T \mathbf{x}_j + b) > 0]$$

因此，相比于使用0/1 loss，在分类问题上研究者倾向于采取近似的，使用概率来解释的平滑loss：

- 模型的输出不再是0/1，而是一个是否为1的 $[0,1]$ 概率值
- $y = 1$ 时， \hat{y} 越接近1，loss越低
- $y = 0$ (not -1 here!) 时， \hat{y} 越接近0，loss越低

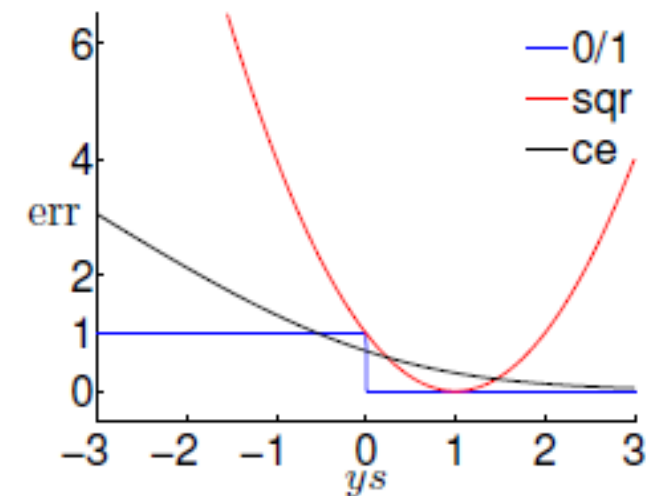
Square loss:

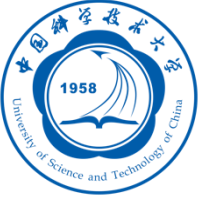
$$L(\mathbf{w}, b) = \sum_{\mathbf{x}_j \in D} (\hat{y}_j - y_j)^2$$

Cross entropy loss:

$$L = -\frac{1}{m} \sum_{\mathbf{x} \in D} \sum_i y_i \ln(\hat{y}_i)$$

分类问题适合cross entropy loss。





代价函数的选择：回归

- 因此，需要对原始输出进行处理，从正负无穷大的区间变成[0,1]的概率值
- 常用的方式：
- 0/1两类问题：Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

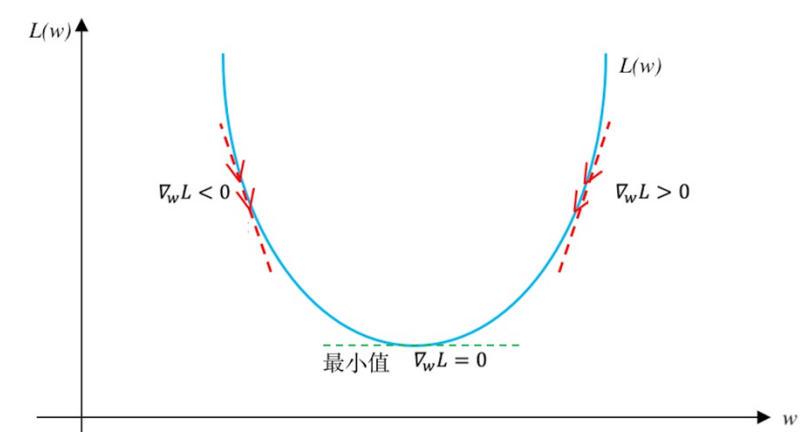
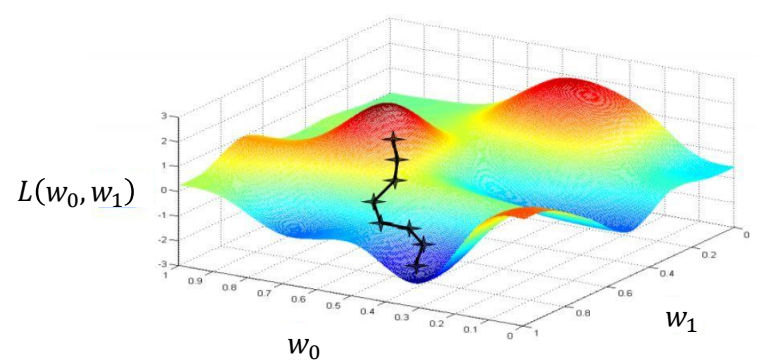
- 多类问题：Softmax function:

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- 注意隐含层激活函数和最后输出层激活函数可能不同！

梯度下降法GD

- 原理：损失函数关于参数 w 的负梯度方向是损失函数下降最快的方向，因此用负梯度方向对参数进行更新



方法	更新梯度	计算梯度的样本	特点
Gradient Descent (GD)	$w \leftarrow w - \eta * \nabla_w L(w)$	全部训练样本	计算复杂度高
Stochastic Gradient Descent (SGD)	$w \leftarrow w - \eta * \nabla_w L_i(w)$	随机抽取一个样本	随机性过大，优化效率低
mini-batch Stochastic Gradient Descent	$w \leftarrow w - \frac{\eta}{p} * \nabla_w \sum_i L_i(w)$	随机抽取的mini-batch样本	计算复杂度低，优化效率高

Mini-batch随机梯度下降法

- mini-batch随机梯度下降法
 - 目前深度学习领域的SGD通常指mini-batch随机梯度下降法
 - 每次迭代随机选取一个mini-batch的样本计算梯度并进行参数更新

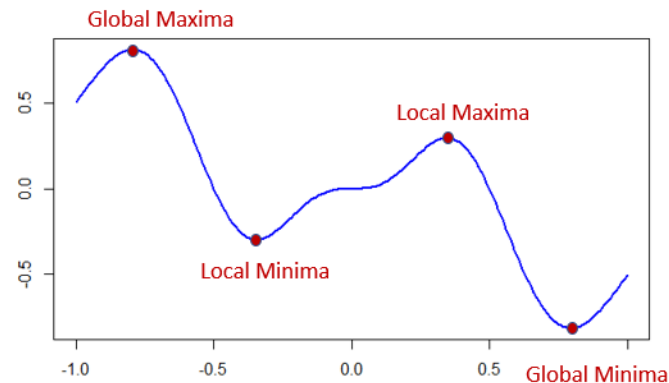
$$g \leftarrow \frac{1}{p} \nabla_w \left(\sum_i L_i(w) \right)$$

$$w \leftarrow w - \eta * g$$

其中 η 是学习率, g 是当前batch的梯度, p 是batch size

- SGD的缺点

- 选择合适的学习率十分困难
- SGD容易收敛到局部最优点, 且可能困在鞍点



动量Momentum

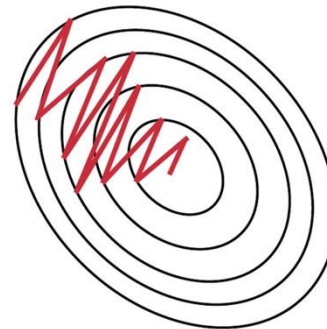
- 带动量的随机梯度下降法
 - 目的：通过积累历史梯度，减小梯度方向的改变，抑制梯度的震荡，加快收敛速度

$$g \leftarrow \frac{1}{p} \nabla_w \left(\sum_i L_i(w) \right)$$

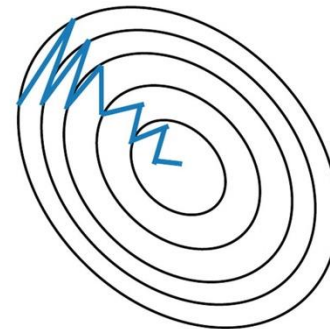
$$v \leftarrow \mu * v - \eta g$$

$$w \leftarrow w + v$$

其中， μ 是动量因子， v 是累计动量



Stochastic Gradient
Descent **without**
Momentum



Stochastic Gradient
Descent **with**
Momentum

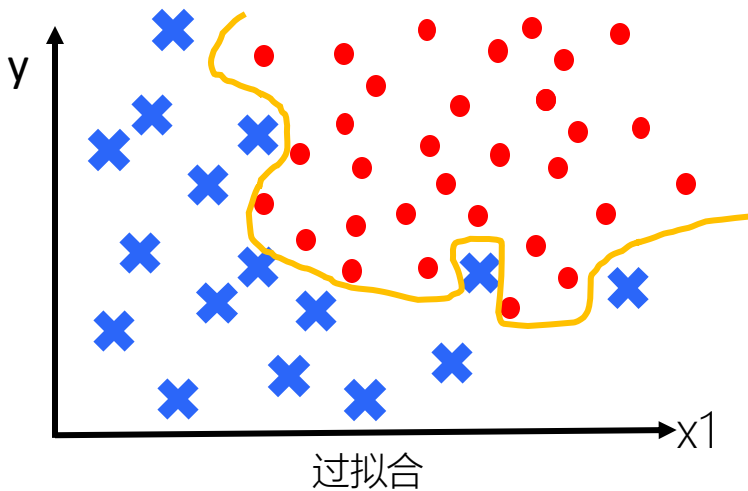
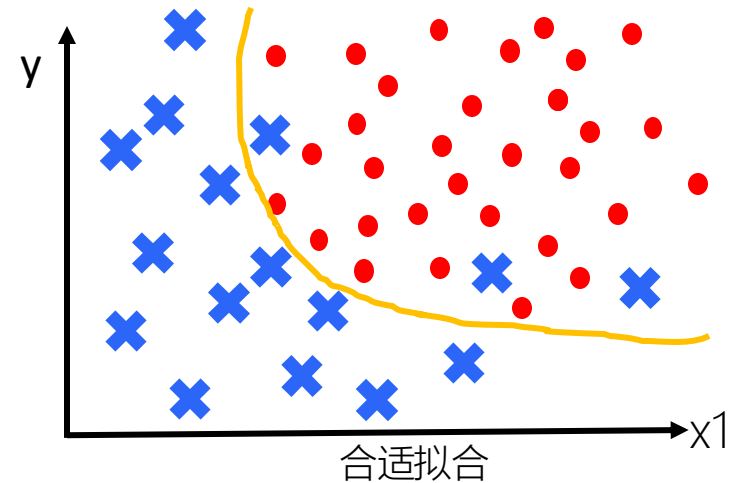
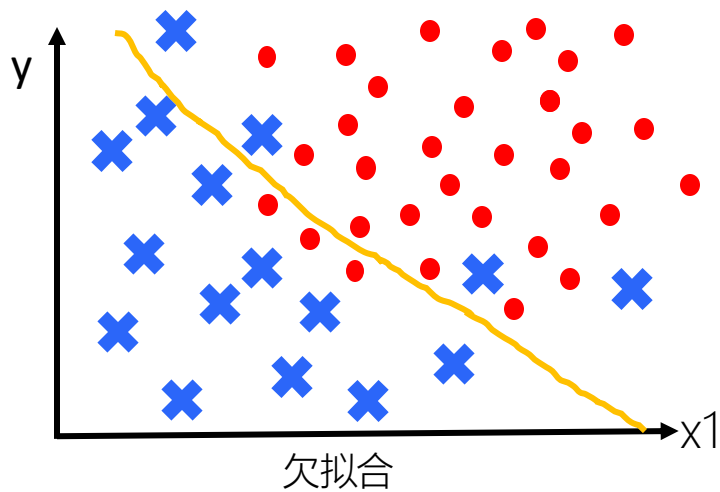
Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature*. 323 (6088): 533–536.



提纲

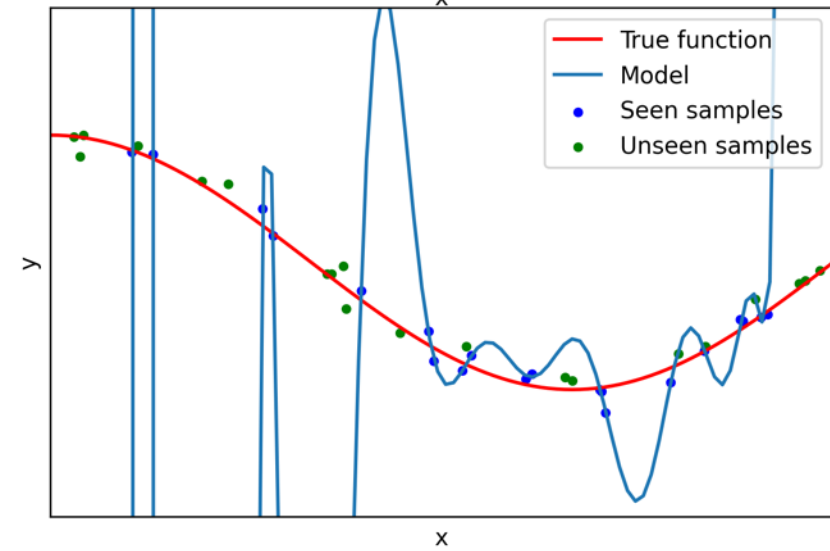
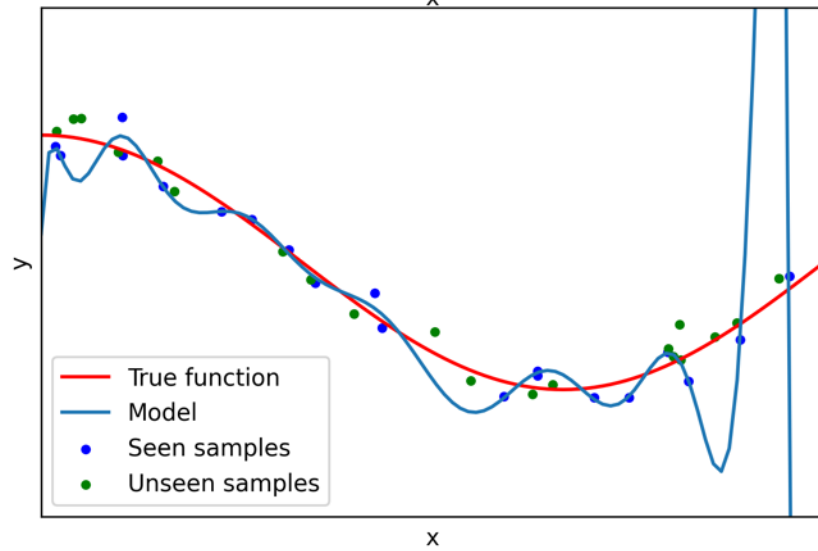
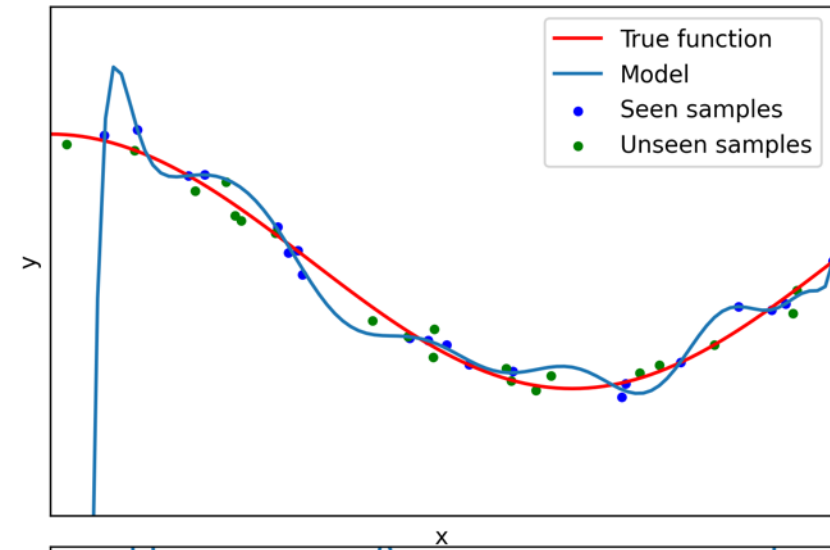
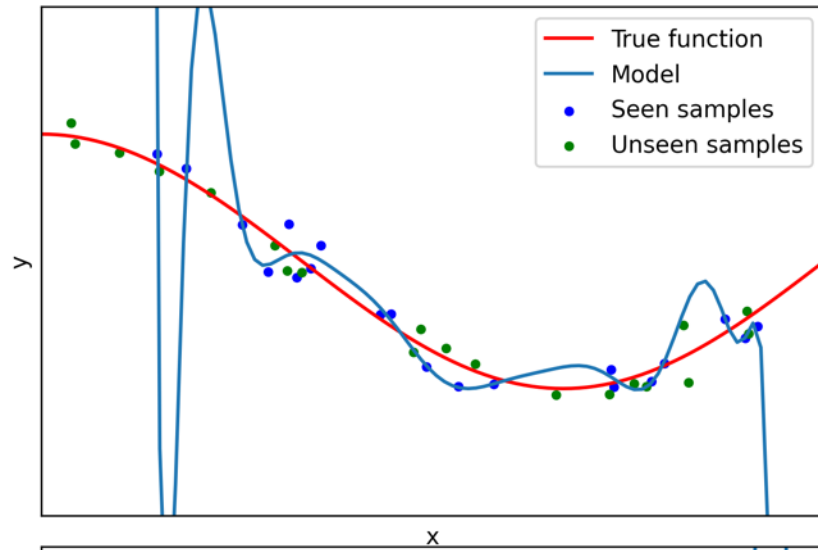
- 机器学习
- 神经网络
- 神经网络训练方法
- 神经网络设计基础
- 过拟合与正则化
- 交叉验证
- 本章小结

欠拟合和过拟合



欠拟合: 训练考虑的维度太少, 拟合函数无法满足训练集, 误差较大。

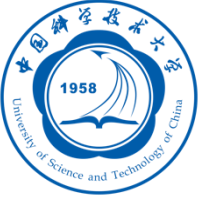
过拟合: 训练考虑的维度太多, 使得拟合的函数很完美的接近训练数据集, 但泛化能力差, 对新数据预测能力不足。





定义

- 机器学习不仅要求模型在训练集上求得一个较小的误差，在测试集上也要表现好。因为模型最终是要部署到没有见过训练数据的真实场景。提升模型在测试集上的预测效果叫做**泛化**。
- **过拟合 (overfitting)** 指模型过度接近训练的数据，模型的泛化能力不足。具体表现为在训练数据集上的误差很低，但在测试数据集上的误差很大。



过拟合

- 神经网络的层数增加，参数也跟着增加，表示能力大幅度增强，极容易出现过拟合现象。
- 主要原因
 - 假设池太大
 - 受到data variance影响明显。
- 1, 2, 3, 4, 5.1, 6, 7, 8, 9....
- 一般人: 10, 11, 12, 13, 14, 15...
- 聪明人: 10.1, 11, 12, 13, 14, 15.1, 16...
- 过于聪明的模型，容易过度解读数据存在的噪声为规律

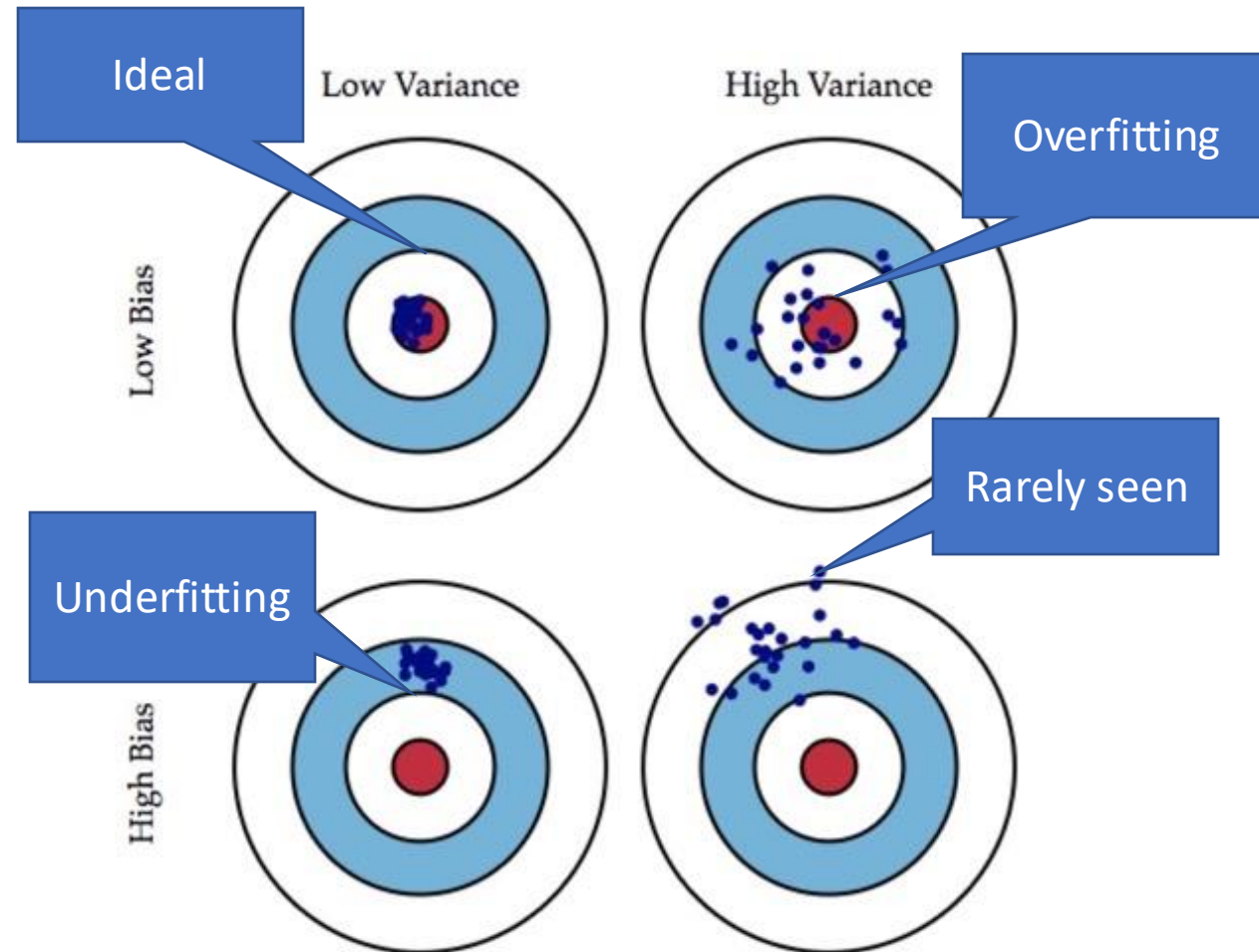


误差的来源: Bias and Variance

- Expect error = $\text{Bias}^2 + \text{Variance} + \text{Noise}$
- Bias: difference between the average prediction of our model and the correct value which we are trying to predict
 - Pay little attention to training data and oversimplifies the model
 - High bias: high error on training and testing set
- Variance: the variability of model prediction for a given data point
 - How much will model varies with different seen data samples
 - Pay a lot of attention to training data and does not generalize on the unseen data
 - High variance: model fits random noise of data, rather than valid information (overfitting)

Visualized Relationship

- Every node is a possible trained model and the bull's eye means the truth model.
- How to train a model with low variance and bias?

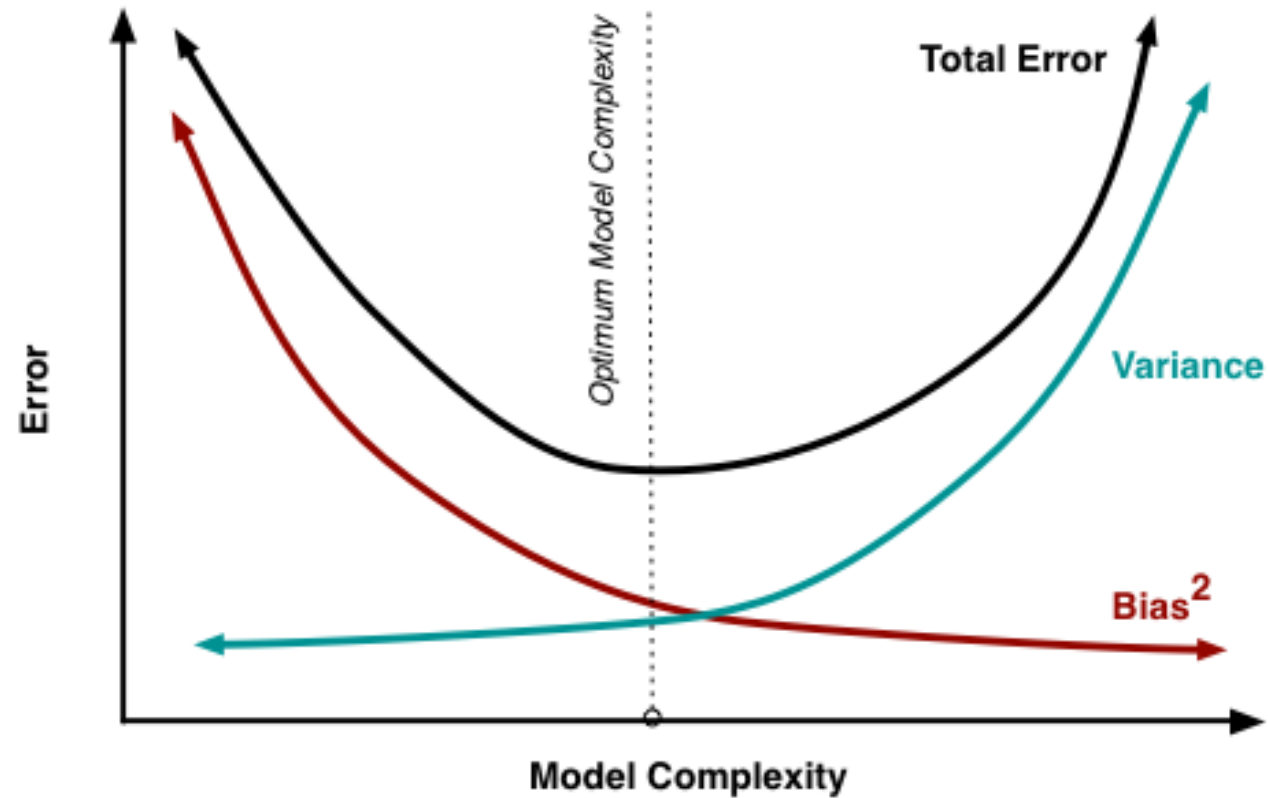




Bias-Variance-Model Tradeoff

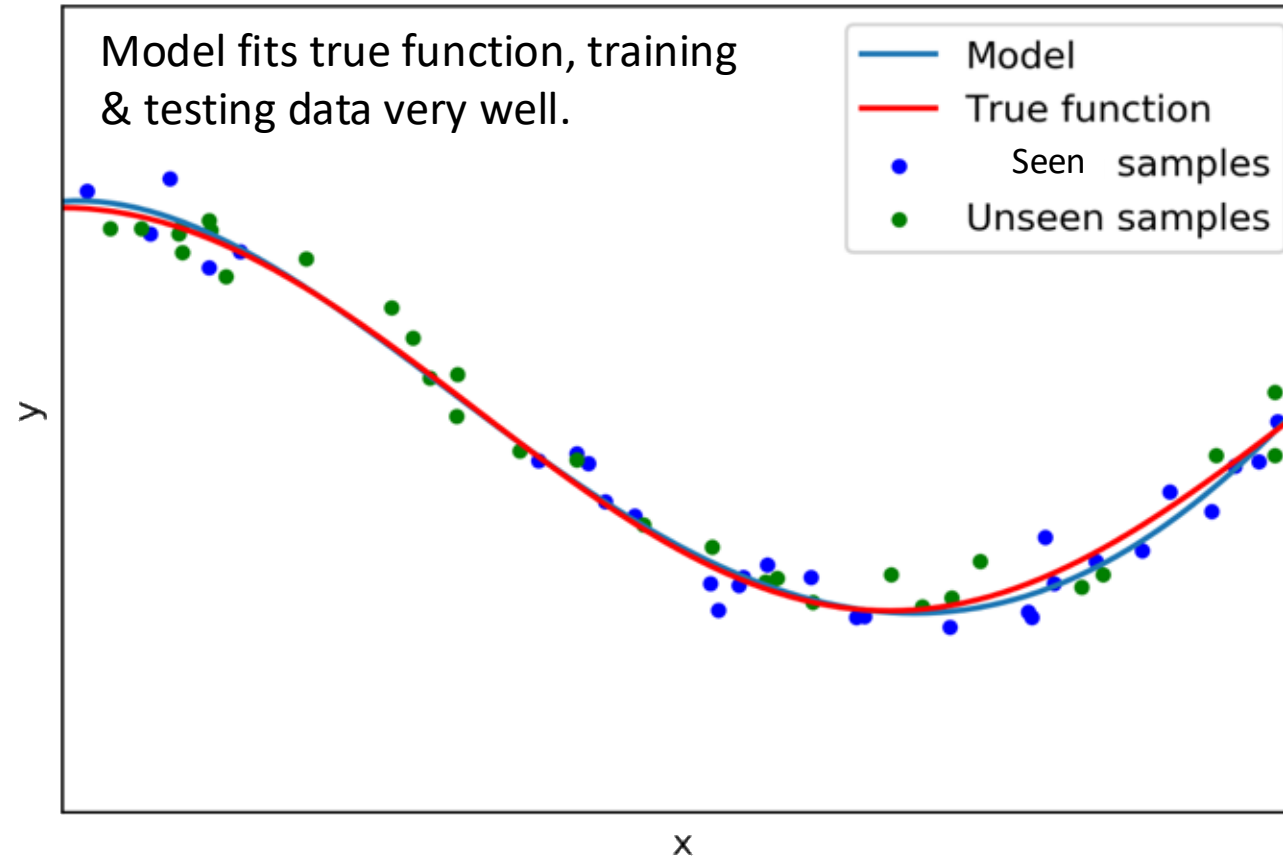
- **Complex** models have low bias and high variance:
 - Low bias: complicated models capture a lot of feature in the training data, perform well on training
 - High variance: testing set may not have the same feature
 - **Overfitting!**
- **Simple** models have low variance and high bias:
 - Just the other way round for complicated models
 - **Underfitting!**

Bias-Variance-Model Tradeoff



Our goal is to select models that are of optimal complexity: not too simple, not too complex

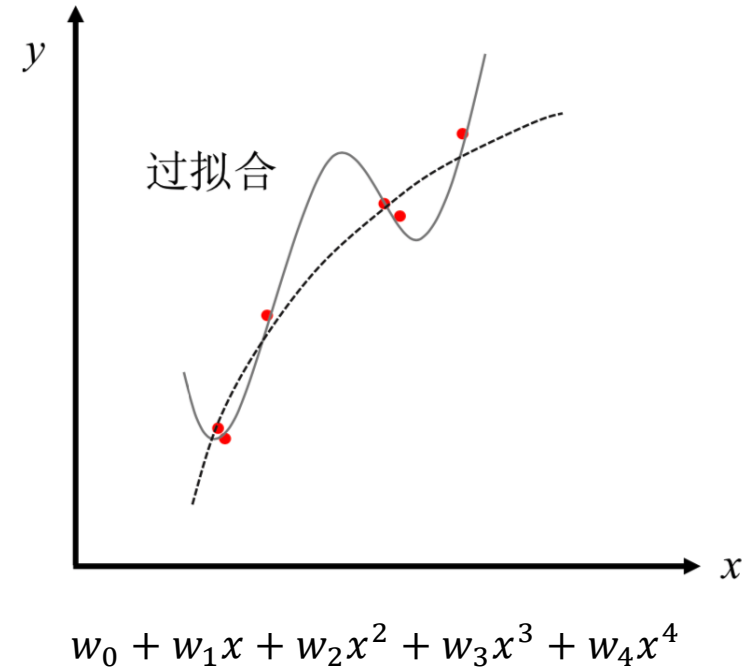
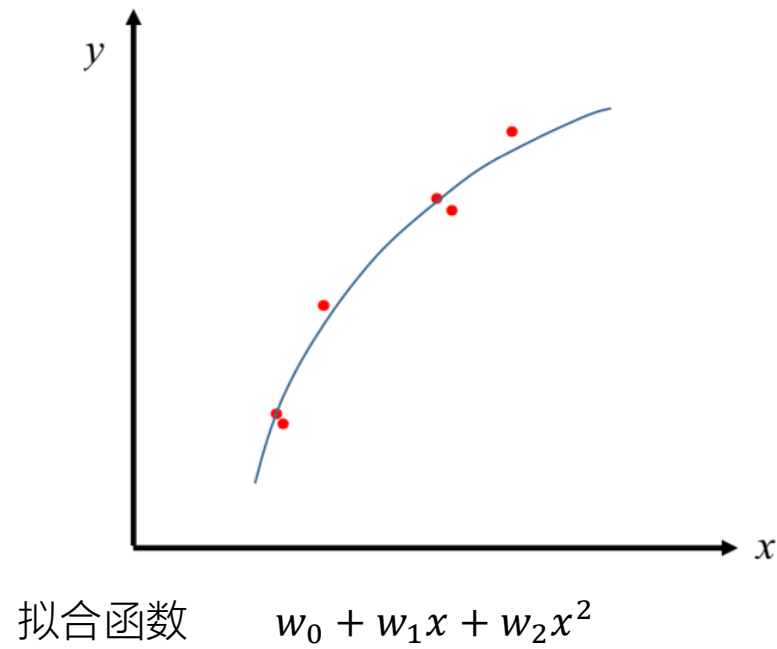
Just Right: Regression with degree=4





- If the bias is high, we can use more complex model
 - But this increases variance
- If the variance is high, we can use less complex model
 - But this increases bias
- 参数范数惩罚、稀疏化、Bagging集成、Dropout、提前终止、数据集扩增等方法可以有效抑制过拟合

正则化思路



$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m \|y_i - \hat{y}_i\|^2$$

目标函数

加上惩罚项使 w_3 、 w_4 足够小

$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$$

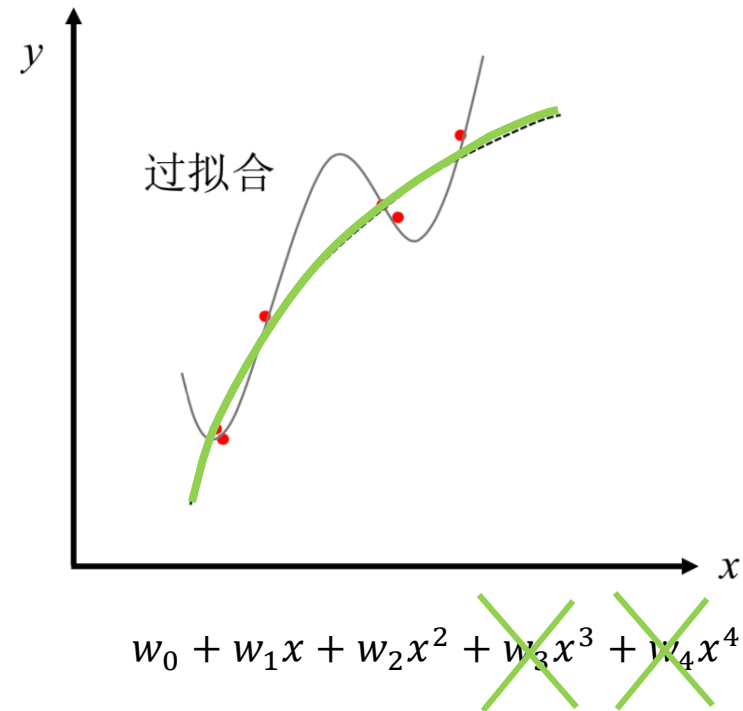


$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 + C_1 * w_3^2 + C_2 * w_4^2$$

C_1 、 C_2 可取常数，如取1000

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 + 1000 w_3^2 + 1000 w_4^2$$

要使目标函数最小，则应有 $w_3 \approx 0$ 、 $w_4 \approx 0$



$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 + C_1 * w_3^2 + C_2 * w_4^2$$



在损失函数中增加一个惩罚项，惩罚高阶参数，使其趋近于0

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 + \theta \sum_{j=1}^k w_j^2$$

正则化项/惩罚项

θ 为正则化参数，神经网络中的参数包括权重 \mathbf{w} 和偏置 \mathbf{b} ，正则化过程仅对权重 \mathbf{w} 进行惩罚，正则化项记为

$$\Omega(\mathbf{w})$$

正则化后的损失函数记为

$$\tilde{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = L(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \theta \Omega(\mathbf{w})$$

L^2 正则化

L^2 正则化项 $\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2$

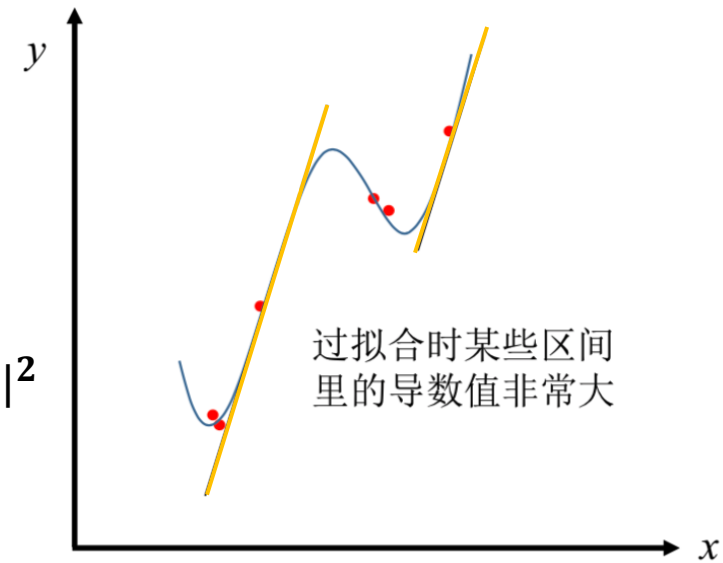
目标函数 $\tilde{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = L(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \frac{\theta}{2} \|\mathbf{w}\|^2$

L^2 正则化如何避免overfitting?

$$\nabla_{\mathbf{w}} \tilde{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \nabla_{\mathbf{w}} L(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \theta \mathbf{w}$$

单步梯度更新权重 $\mathbf{w} \leftarrow \mathbf{w} - \eta (\nabla_{\mathbf{w}} L(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \theta \mathbf{w})$

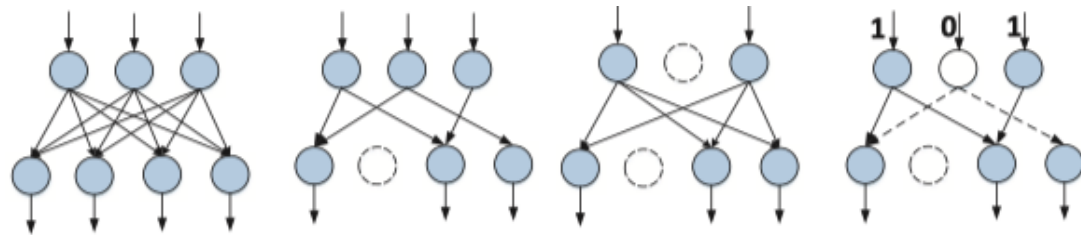
$$\mathbf{w} \leftarrow (1 - \eta\theta) \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}; \mathbf{X}, \mathbf{y})$$



通过 L^2 正则化后, \mathbf{w} 权重值变小, 网络的复杂度降低, 对数据拟合的也更好。

稀疏化

- 训练时让网络中的很多权重或神经元为0
- 90%的权重或神经元为0
- 降低正向传播时的计算量

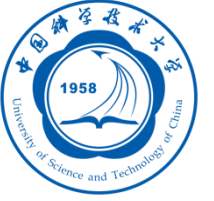


稠密MLP

突触稀疏

神经元稀疏

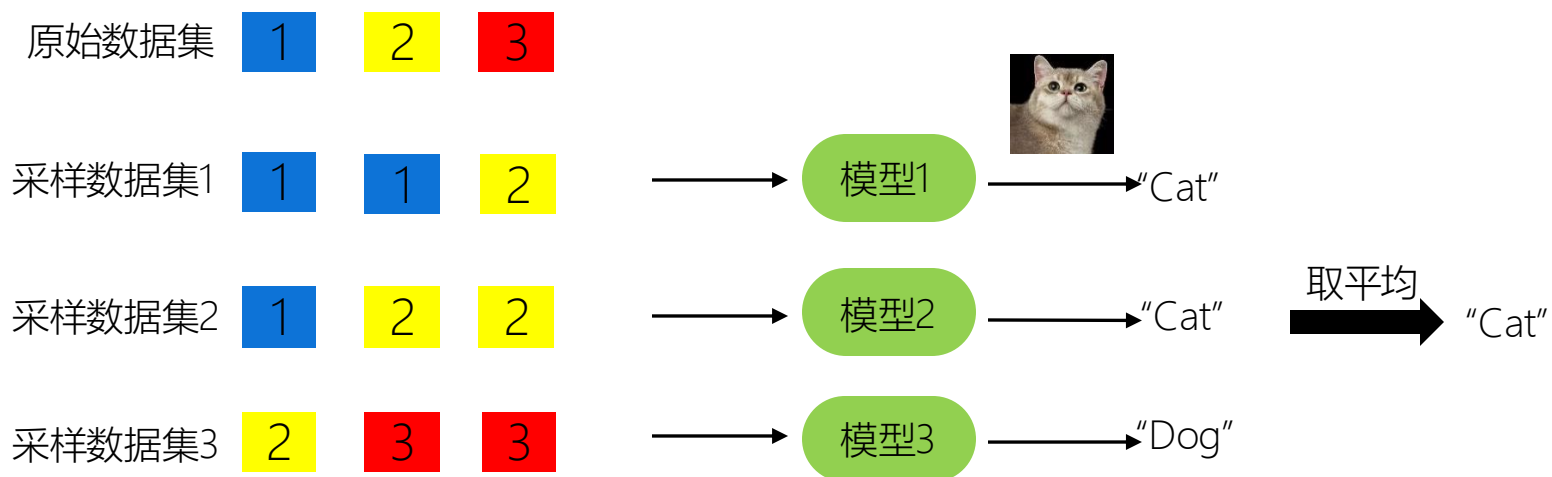
动态稀疏



Bagging集成方法

- Bagging 训练不同的模型来共同决策测试样例的输出，不同的模型即使在同一个训练数据集上也会产生不同的误差。
- Bagging可以多次重复使用同一个模型、训练算法和目标函数进行训练。
- Bagging的数据集从原始数据集中重复采样获取，数据集大小与原始数据集保持一致。

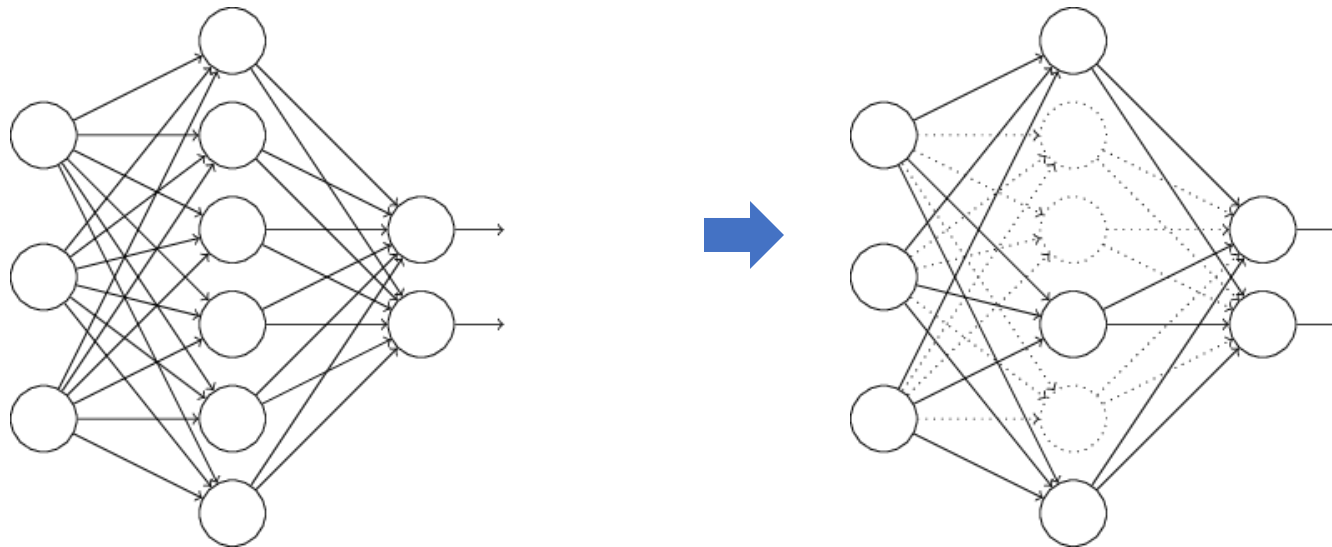
假设集成 $k=3$ 个网络模型



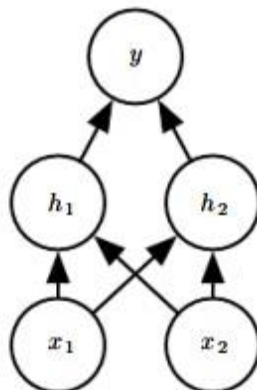
➤ 模型平均是减小泛化误差的一种可靠方法

Dropout

- L^2 和 L^1 正则化是通过在目标函数中增加一项惩罚项，Dropout正则化是通过在训练时暂时修改神经网络来实现的。
- Dropout思路：在训练过程中随机地“删除”一些隐层单元，在计算时无视这些连接。
- 实际使用的时候使用所有神经元。

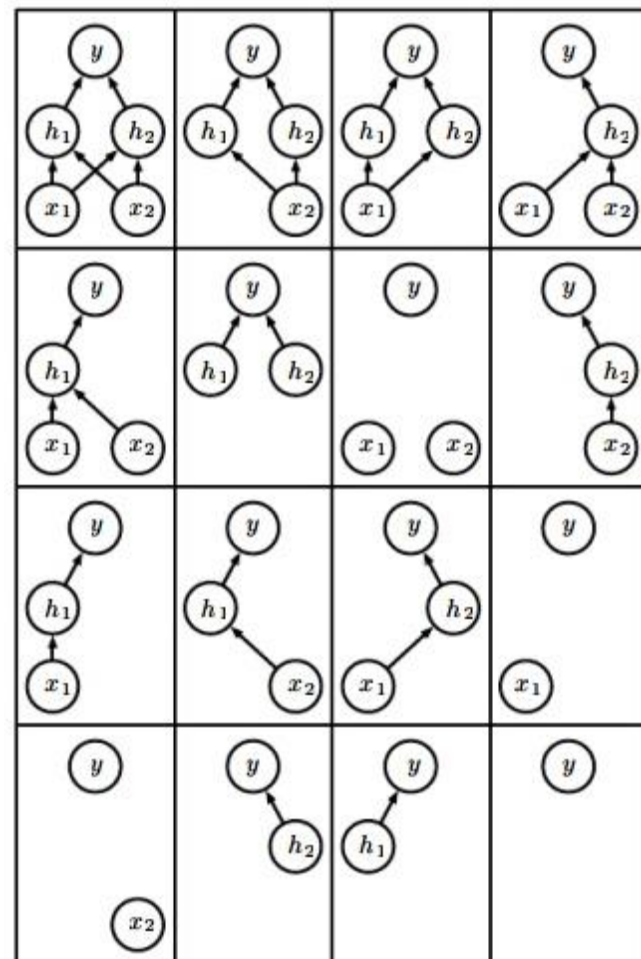
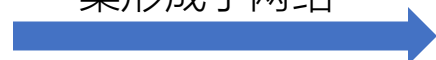


乘零的Dropout 算法

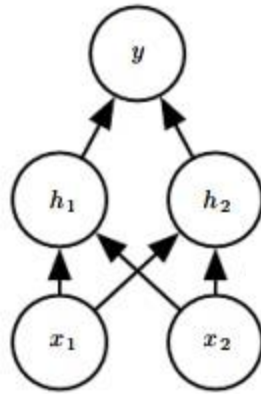


基础网络

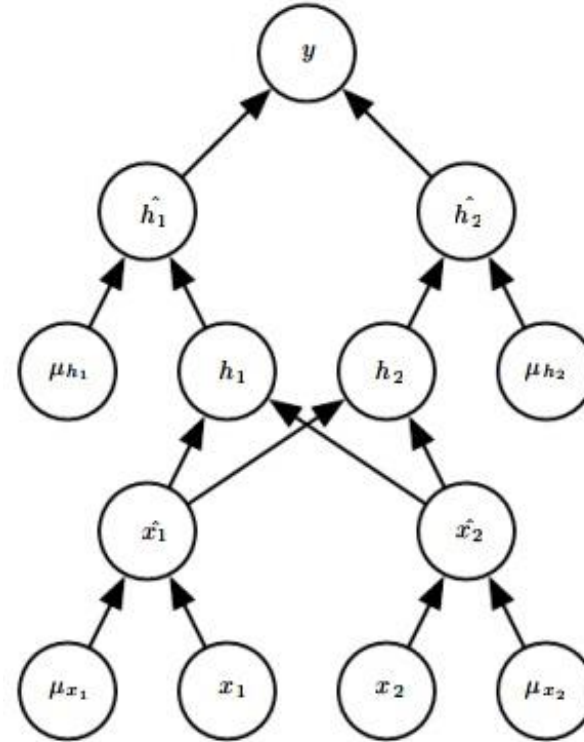
从基础网络中丢
弃不同的单元子
集形成子网络



子网络集成



基础网络



- 随机对掩码 μ 进行采样，输入单元的采样概率为0.8，隐藏单元的采样概率为0.5。网络中的每个单元乘以相应的掩码后沿着网络的其余部分继续向前传播



其他方法

- 提前终止
 - 当训练较大的网络模型时，能够观察到训练误差会随着时间的推移降低但测试集的误差会再次上升。因此，在训练过程中返回预定迭代次数内测试误差达最低的参数设置，这种策略称之为提前终止。
- 多任务学习
 - 多任务学习通过多个相关任务的同时学习来减少神经网络的泛化误差。
- 数据集增强
 - 使用更多的数据进行训练，可对原数据集进行变换形成新数据集添加到训练数据中。
- 参数共享
 - 强迫两个模型（监督模式下的训练模型和无监督模式下的训练模型）的某些参数相等，使其共享唯一的一组参数。



提纲

- 机器学习
- 神经网络
- 神经网络训练方法
- 神经网络设计基础
- 过拟合与正则化
- 交叉验证
- 本章小结



交叉验证

- 传统机器学习中将数据集划分为测试集和训练集
 - 利用训练集进行训练
 - 利用测试集评估算法效果
- 划分测试集的目的
 - 避免过拟合
 - 评判各模型的鲁棒性。



交叉验证

- 深度学习的数据集划分：训练集、验证集、测试集
 - 利用训练集训练模型参数（权重和偏置）
 - 利用验证集确定神经网络超参数（网络结构、学习率、训练迭代次数等）
 - 利用测试集评估模型效果

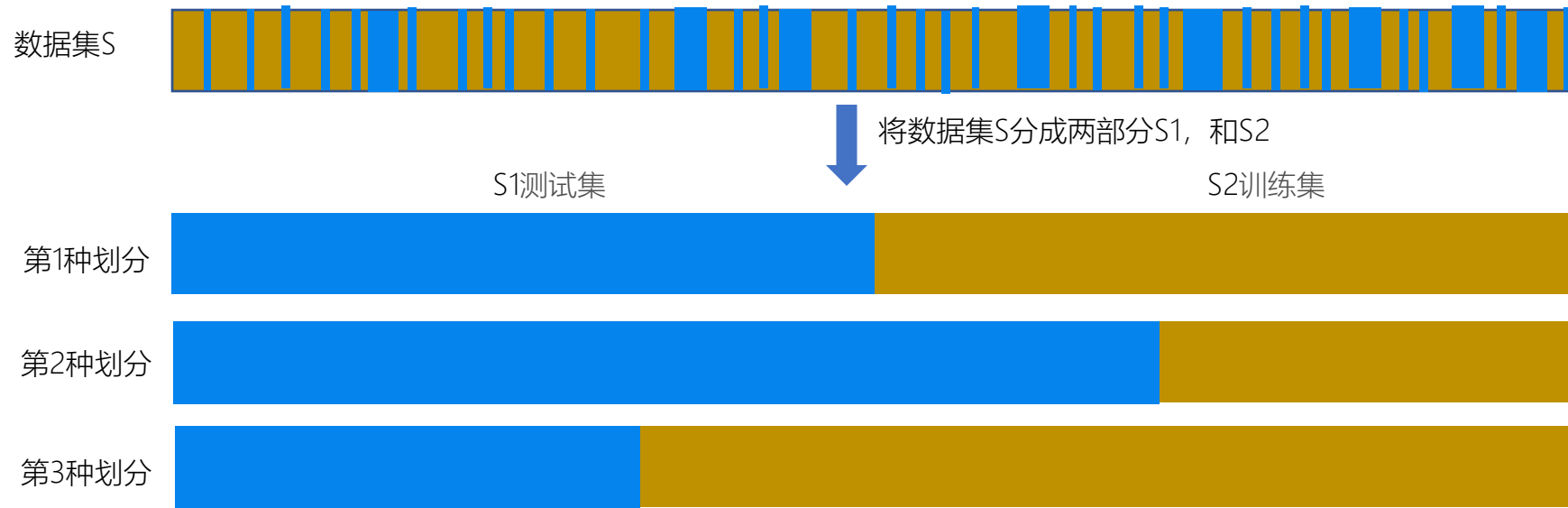


交叉验证

- 交叉验证的方式给每个样本作为测试集和训练集的机会，充分利用样本信息，保证了鲁棒性，防止过拟合。
- 选择多种模型进行训练时，使用交叉验证能够评判各模型的鲁棒性。

最简单的验证方式

测试集和训练集

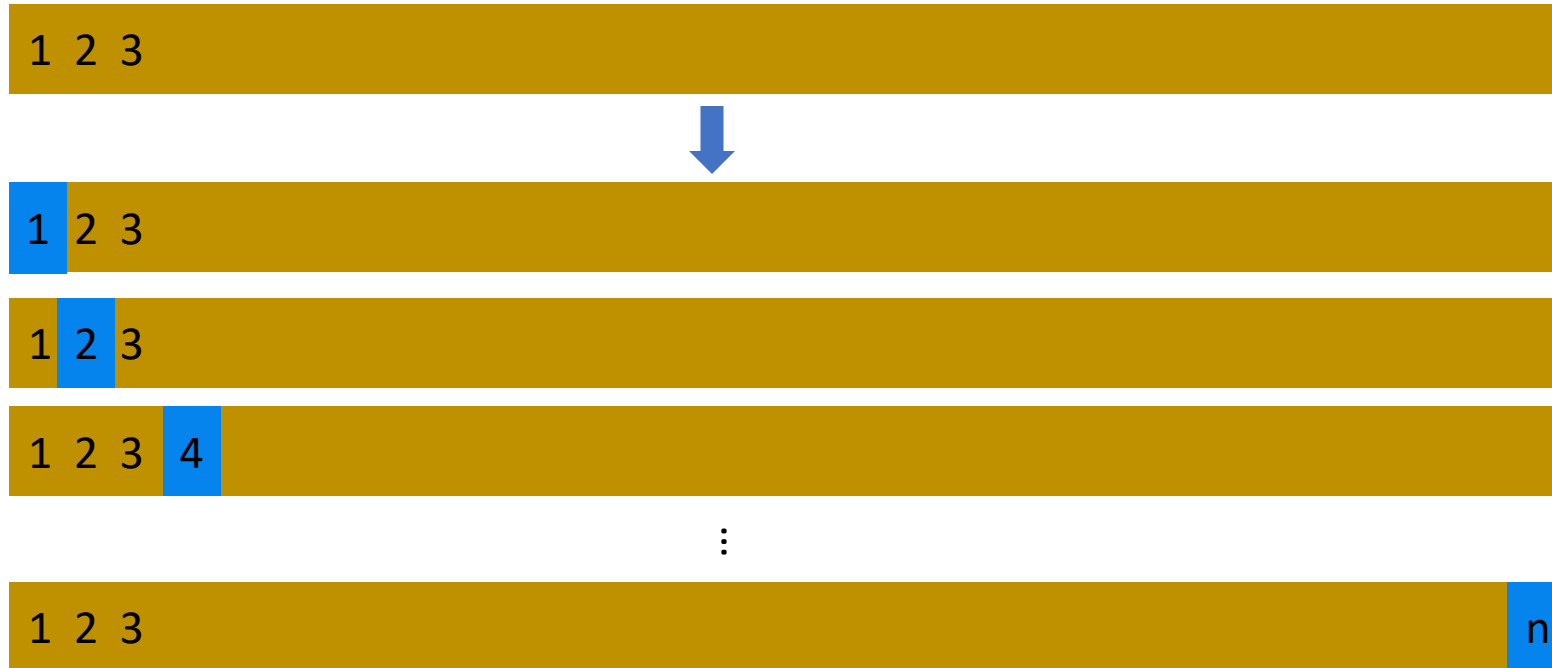


➤ 不同划分方式下，得到的评估结果变动较大

缺点：最终模型与参数的选取将极大程度依赖于对训练集和测试集的划分方法
只有部分数据参与了模型的训练

Leave-one-out cross-validation验证方法

数据集S包
含n个数据



每次取出一个数据作为测试集的唯一元素，而其他 $n-1$ 个数据都作为训练集用于训练模型和调参。最终训练出 n 个模型，得到 n 个评估结果（e.g.MSE）。将这 n 个评估结果取平均得到最终的测试集评估结果。

缺点：计算量过大，耗费时间长

K-折交叉验证 (k-fold cross validation)

数据集S包
含n个数据
分成K=5份



不重复地每次取其中一份做测试集，用其他K-1份做训练集训练模型，之后计算该模型在测试集上的 MSE_i ，最后再将K次的 MSE_i 取平均得到最后的MSE

$$MSE = \frac{1}{K} \sum_{i=1}^K MSE_i$$

Leave-one-out cross-validation是一种特殊的K-fold Cross Validation (K=n)

优点：所有的样本都被作为了训练集和测试集，每个样本都被验证一次，相比Leave-one-out cross-validation，计算成本降低，耗时减少



深度学习与交叉验证

- 在图像分类、目标检测、机器翻译等大部分应用领域，通常可以收集到较大规模的数据集，因此无需使用K-折交叉验证。
- 在某些领域，如医学图像处理、遥感图像处理等，数据收集较为困难，依然需要使用K-折交叉验证来评估模型的泛化能力，避免神经网络过拟合。



小结

➤机器学习

掌握单变量线性回归、多变量线性回归。

➤神经网络

掌握感知机、神经元、激活函数、偏置等的概念和相关用法。

➤神经网络训练方法

了解数据正向传播和反向传播的过程。

➤神经网络设计基础

了解常用激活函数的优缺点，损失函数的作用和种类、掌握至少一种验证方式。

➤过拟合与正则化

了解过拟合出现的原因并掌握防止过拟合的正则化方法。

➤交叉验证

掌握至少一种交叉验证的原理和方法。