



北京交通大学《深度学习》课件

# 第二讲 基础知识

北京交通大学 《深度学习》课程组





## 2.1 机器学习基本概念

## 2.2 数学基础

## 2.3 线性模型

- Logistic回归
- Softmax回归
- 感知机

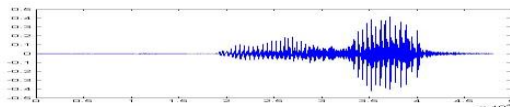


## 2.1 机器学习基础

### ■ 机器学习 $\approx$ 构建一个映射函数

- 语音识别

$$f(\text{语音波形}) = \text{"你好"}$$



- 图像识别

$$f(\text{猫咪照片}) = \text{"猫"}$$



- 围棋

$$f(\text{棋局状态}) = \text{"5-5" (落子位置)}$$



- 对话系统

$$f(\text{"你好"}) = \text{"今天天气真不错"}$$

用户输入

机器回应



# 机器学习的关键要素

## ➤ 模型

- 线性方法:  $f(\mathbf{x}, \theta) = \mathbf{w}^T \mathbf{x} + b$
- 广义非线性方法:  $f(\mathbf{x}, \theta) = \mathbf{w}^T \phi(\mathbf{x}) + b$ 
  - ✓ 如果  $\phi(\mathbf{x})$  为可学习的非线性基函数,  $f(\mathbf{x}, \theta)$  就等价于神经网络。

## ➤ 学习准则

- 期望风险（期望误差） —— 衡量模型的好坏

$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} [\mathcal{L}(f(\mathbf{x}), y)]$$

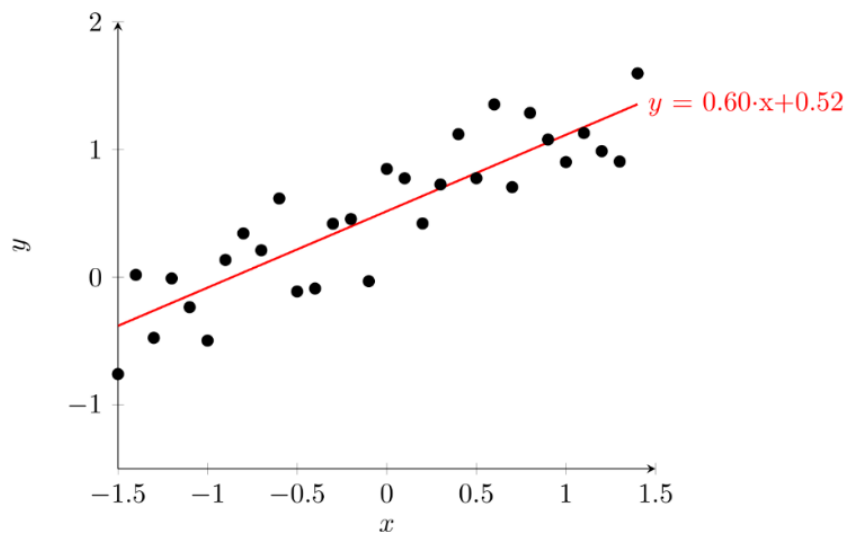
- ✓  $\mathcal{L}(f(\mathbf{x}), y)$ : 损失函数, 量化模型预测和真实标签之间的差异
- ✓ 一个好的模型应该有一个较小的期望风险（风险最小化准则）

## ➤ 优化算法

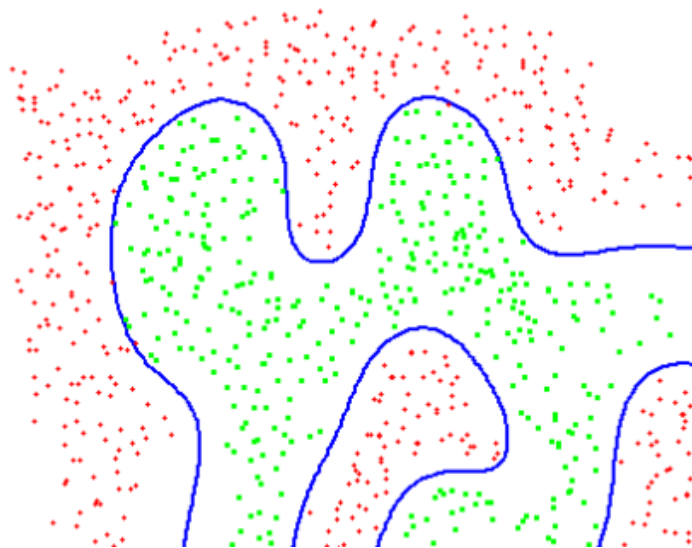
- 梯度下降



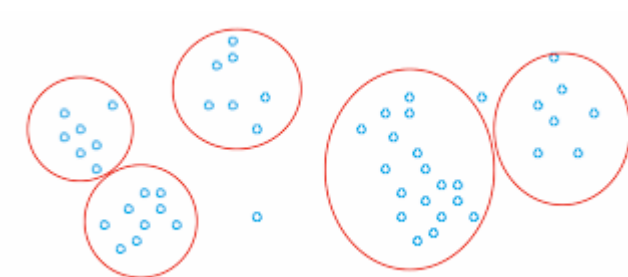
# 常见的机器学习问题



回归  
(Regression)



分类  
(Classification)



聚类  
(Clustering)

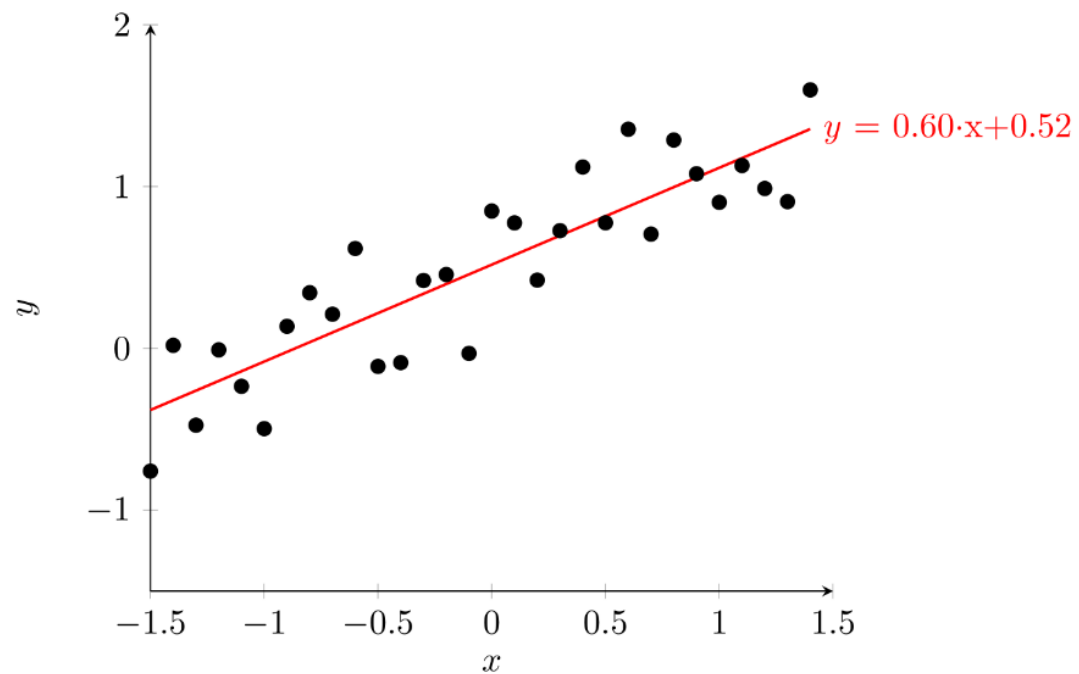


# 机器学习的简单示例

## ➤ 以线性回归（Linear Regression）为例

### • 模型：

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T \mathbf{x} + b$$



如何确定  $w$  和  $b$  ?



## ➤ 以线性回归 (Linear Regression) 为例

- **学习准则：** 关键是如何定义损失函数

✓ **0-1损失函数** 
$$\mathcal{L}(y, f(x, \theta)) = \begin{cases} 0 & \text{if } y = f(x, \theta) \\ 1 & \text{if } y \neq f(x, \theta) \end{cases}$$

- 数学性质不好：不连续且导数为0，难以优化

✓ **平方损失函数** 
$$\mathcal{L}(y, \hat{y}) = (y - f(x, \theta))^2$$



## ➤ 以线性回归 (Linear Regression) 为例

### • 学习准则：风险最小化准则

✓ 真实数据分布未知，期望风险不可计算，常用经验风险近似

- 给定数据集  $\mathcal{D} = \{x^{(n)}, y^{(n)}\}$ ,  $n \in [1, N]$ , 经验风险为：

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$

✓ 经验风险最小化

- 寻找一组最优的参数  $\theta^*$ ，使经验风险函数最小化

$$\theta^* = \arg \min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

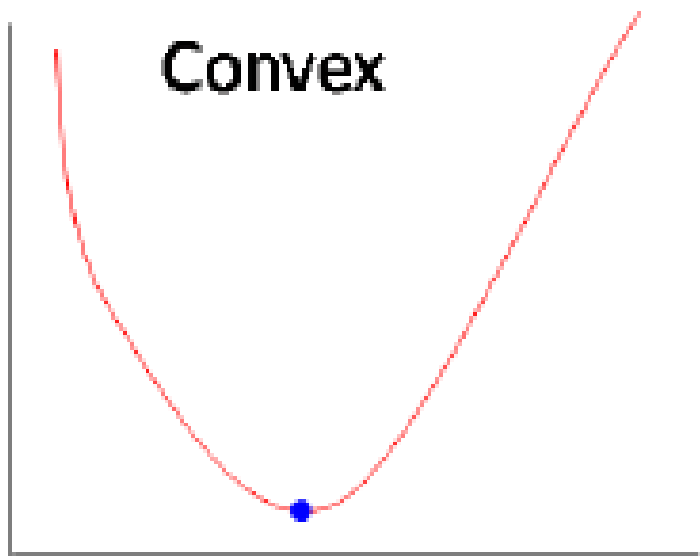
✓ 机器学习问题转化成为一个最优化问题





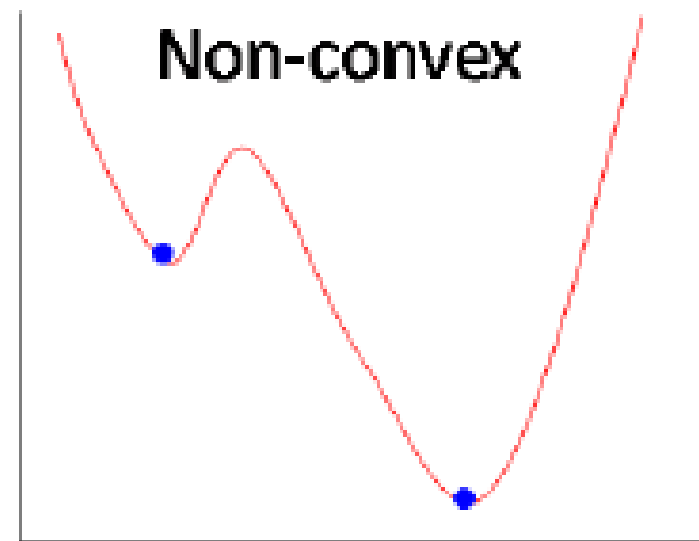
# 最优化问题

- 部分模型（如线性回归）的优化目标是凸函数



$$\min_{\mathbf{x}} f(\mathbf{x})$$

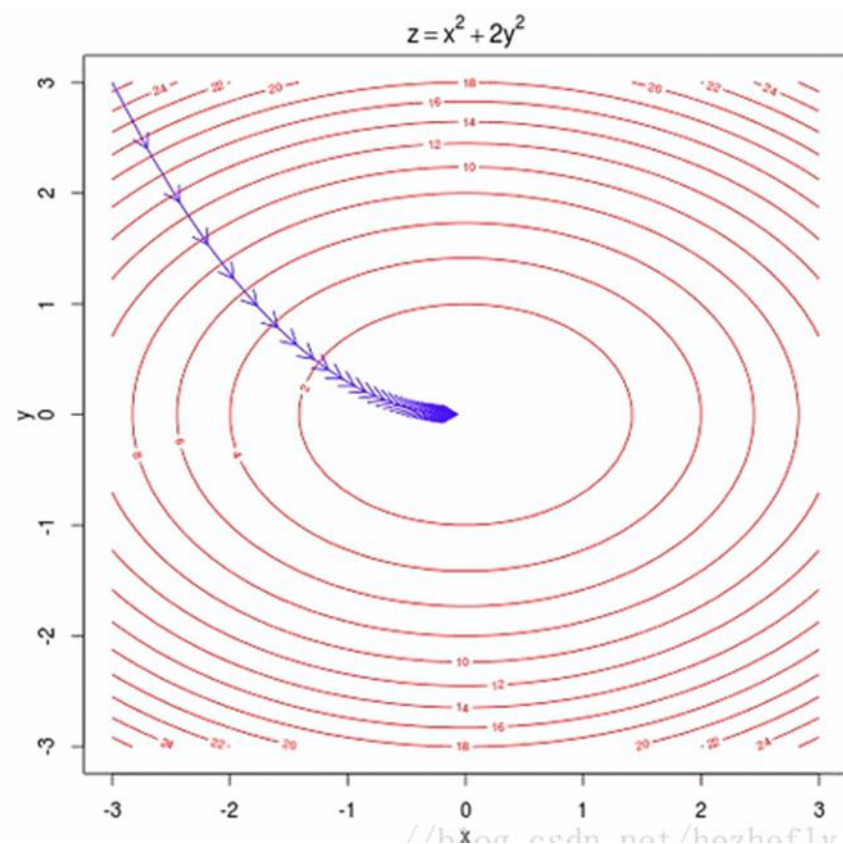
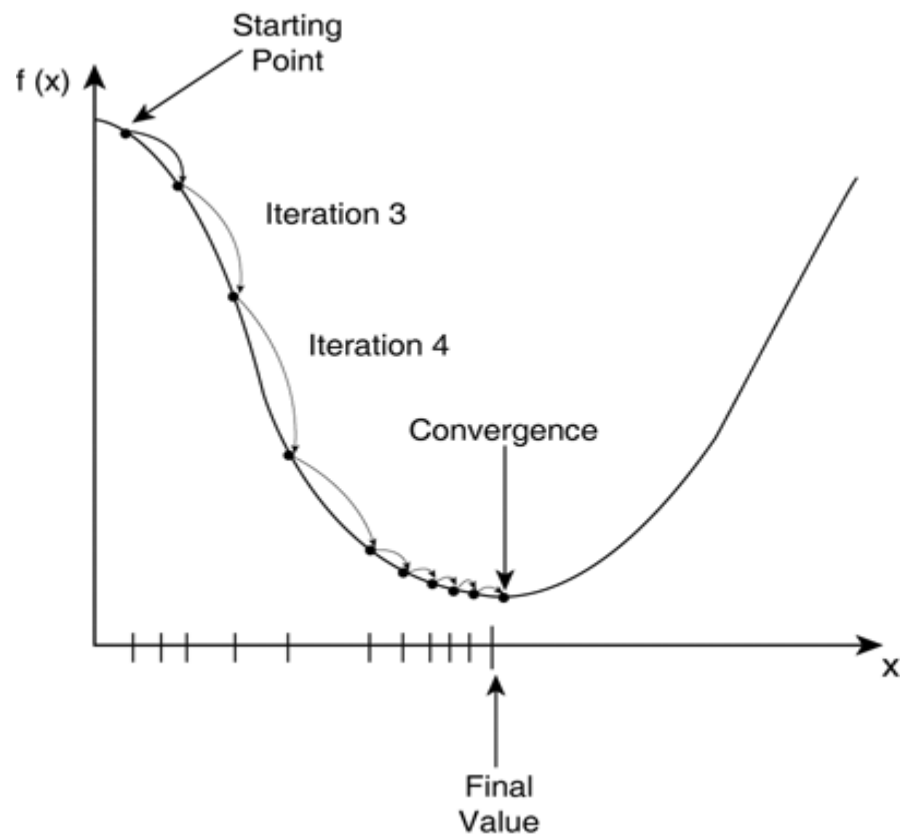
- 部分模型（例如神经网络）的优化目标是非凸的



局部最优解



# 梯度下降法 ( Gradient Descent )



- 经过迭代计算风险函数的最小值

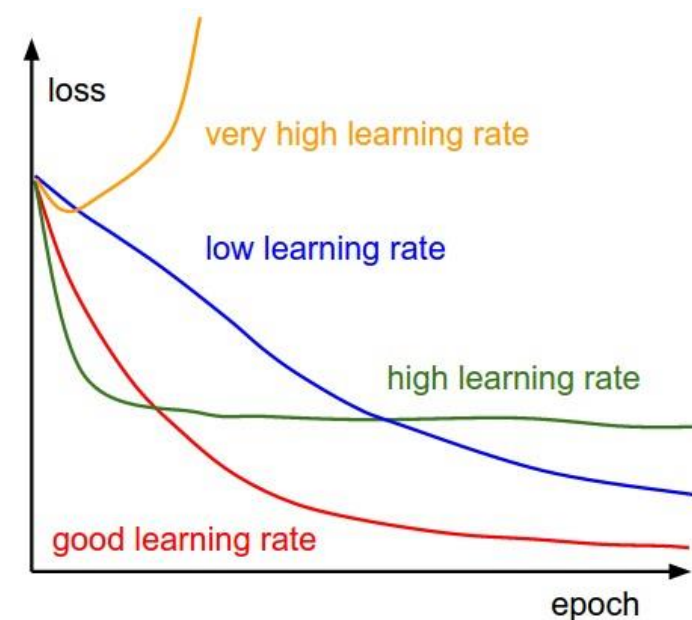
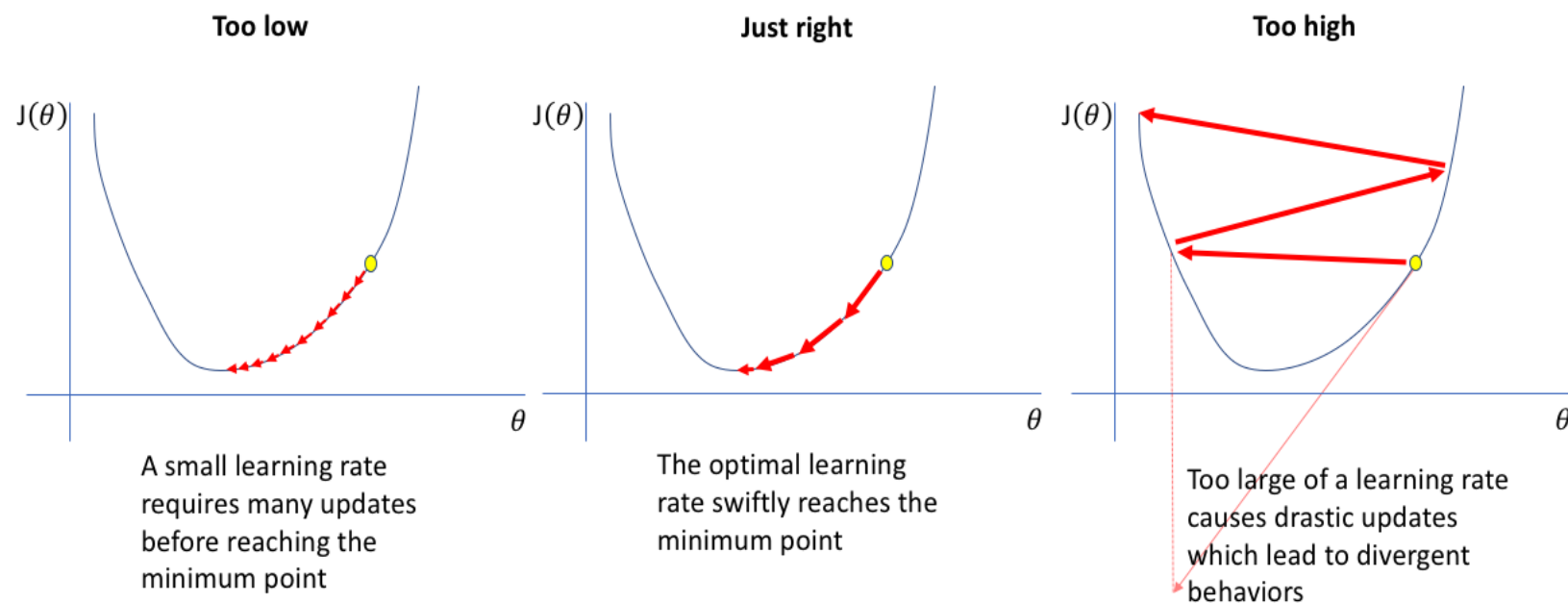
$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \\ &= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}.\end{aligned}$$

搜索步长  $\alpha$  中也叫作**学习率** (Learning Rate)



# 梯度下降法 ( Gradient Descent )

## ■ 学习率是十分重要的超参数!





# 随机梯度下降法 (Stochastic Gradient Descent, SGD)

■ 随机梯度下降法，也叫增量梯度下降，**每次随机用一个样本都进行更新**

✓ 批量梯度下降更新公式：

$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \\ &= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}\end{aligned}$$

✓ 随机梯度下降更新公式：

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta}$$

## 算法 2.1: 随机梯度下降法

输入: 训练集  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ , 验证集  $\mathcal{V}$ , 学习率  $\alpha$

```
1 随机初始化  $\theta$ ;  
2 repeat  
3   对训练集  $\mathcal{D}$  中的样本随机重排序;  
4   for  $n = 1 \cdots N$  do  
5     从训练集  $\mathcal{D}$  中选取样本  $(\mathbf{x}^{(n)}, y^{(n)})$ ;  
6     // 更新参数  
7      $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta; x^{(n)}, y^{(n)})}{\partial \theta}$ ;  
8   end  
9 until 模型  $f(\mathbf{x}; \theta)$  在验证集  $\mathcal{V}$  上的错误率不再下降;  
输出:  $\theta$ 
```

✓ 小批量 (mini-batch) 梯度下降：随机选取一部分训练样本



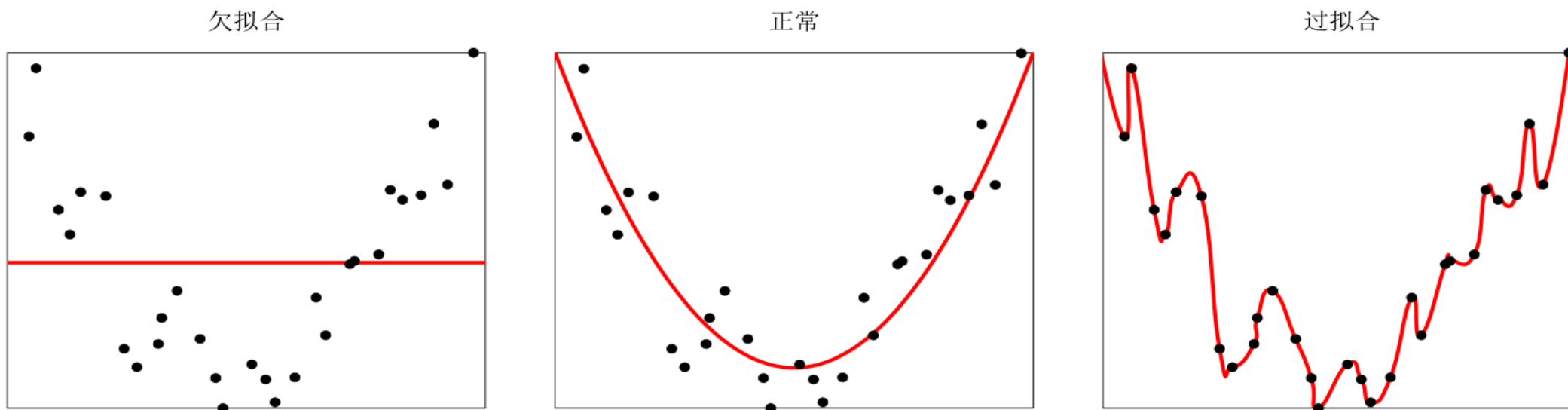
# 欠拟合与过拟合

## ■ 欠拟合 (Underfitting)

- ▶ 对训练样本的一般性质尚未学好

## ■ 过拟合 (Overfitting)

- ▶ 学习器把训练样本学习得“太好”，将训练样本本身的特点当做所有样本的一般性质，导致泛化性能下降。



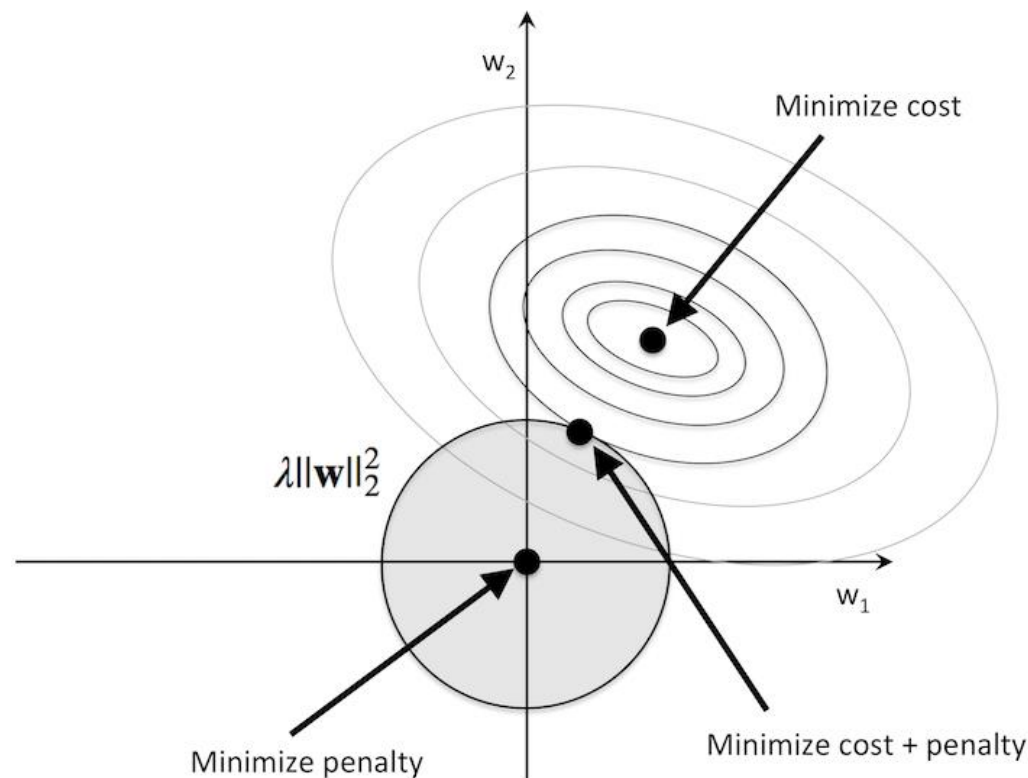


# 正则化 (regularization)

- 为了解决过拟合问题，在经验风险最小化的基础上再引入参数的**正则化 (Regularization)**来限制模型能力，使其不要过度最小化经验风险，这就是**结构风险最小化 (Structure Risk Minimization, SRM) 准则**。

$$\arg \min_{\theta} \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y^{(n)}, f(x^{(n)}; \theta)) + \frac{1}{2} \lambda \|\theta\|^2$$

$L_2$ 范数正则项





# 偏差-方差分解 (Bias-Variance Decomposition)

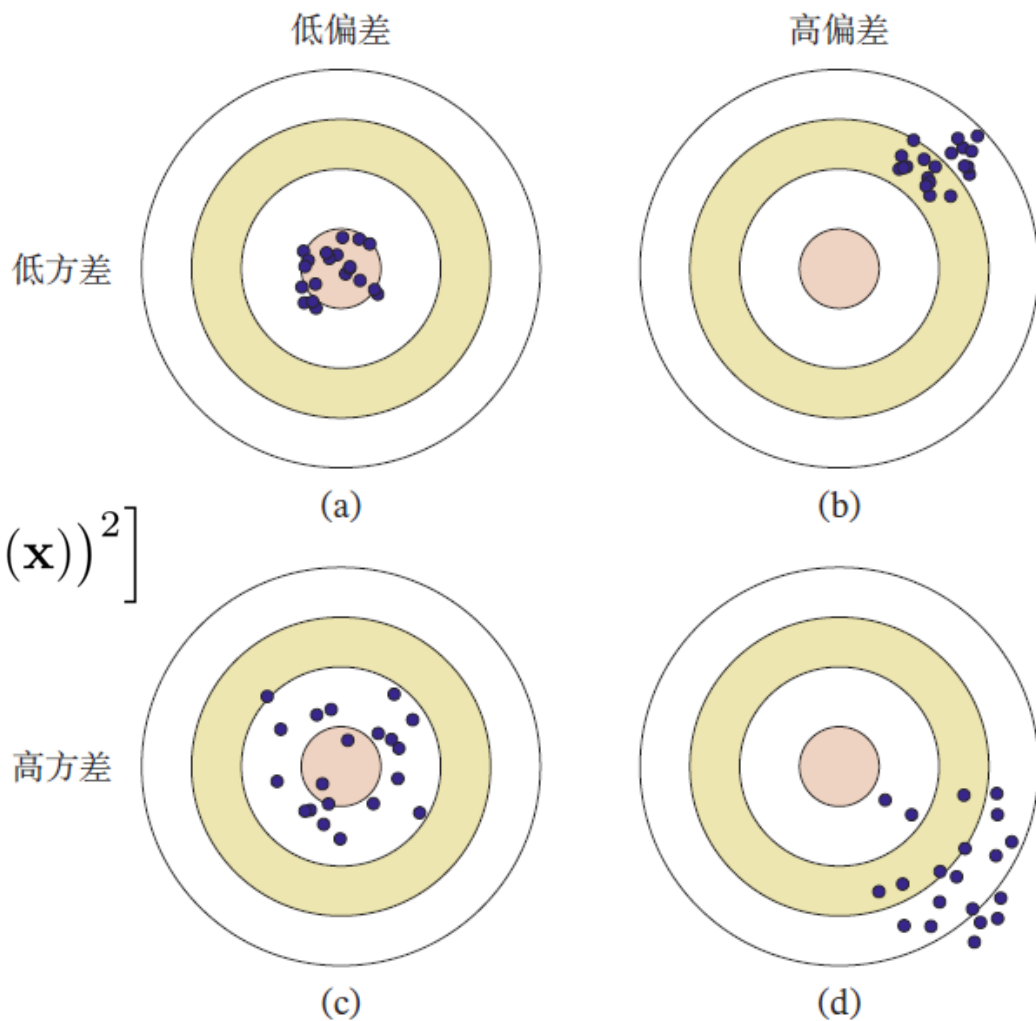
## ■ 最小化期望错误等价于最小化偏差和方差之和

$$\mathcal{R}(f) = (\text{bias})^2 + \text{variance} + \varepsilon.$$

$$\mathbb{E}_{\mathbf{x}} \left[ \left( \mathbb{E}_{\mathcal{D}} [f_{\mathcal{D}}(\mathbf{x})] - f^*(\mathbf{x}) \right)^2 \right]$$

$$\mathbb{E}_{(\mathbf{x}, y) \sim p_r(\mathbf{x}, y)} \left[ \left( y - f^*(\mathbf{x}) \right)^2 \right]$$

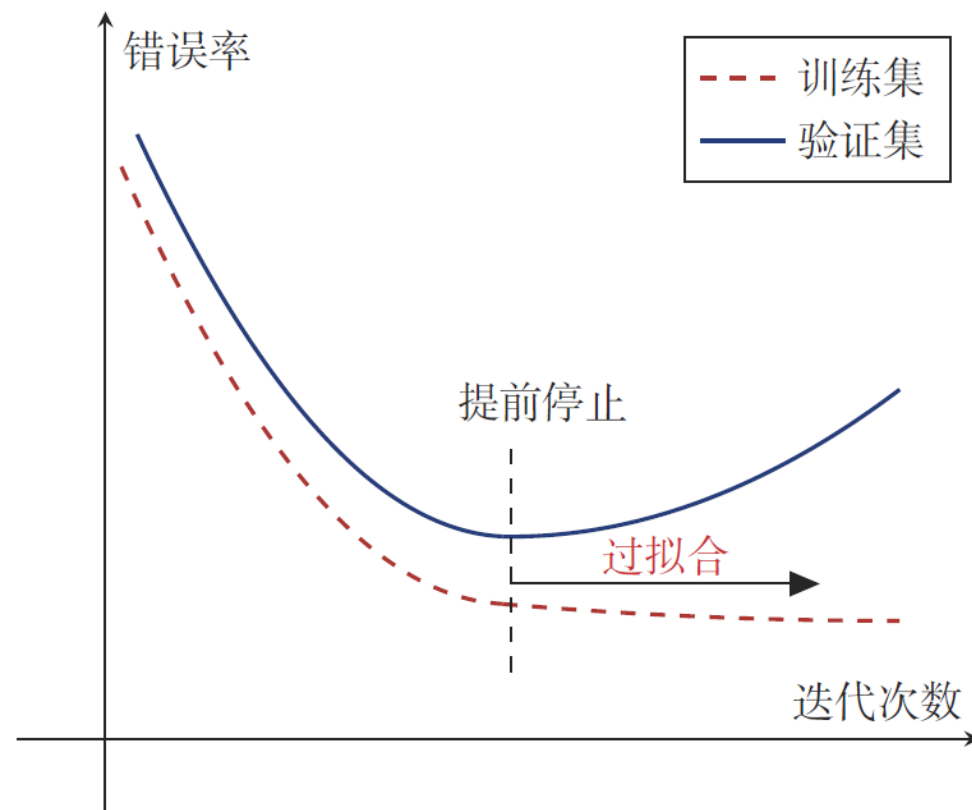
$$\mathbb{E}_{\mathbf{x}} \left[ \mathbb{E}_{\mathcal{D}} \left[ \left( f_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}} [f_{\mathcal{D}}(\mathbf{x})] \right)^2 \right] \right]$$





# 提前停止 (Early Stop)

- 除了训练集和测试集之外，使用一个**验证集 (Validation Set)** 来进行模型选择。在每次迭代时，把新得到的模型 $f(x; \theta)$ 在验证集上计算错误率，如果在验证集上的错误率不再下降，就停止迭代。







## 2.1 机器学习基本概念

## 2.2 数学基础

## 2.3 线性模型

- Logistic回归
- Softmax回归
- 感知机



## ➤ 线性代数

- 向量、矩阵、矩阵运算

## ➤ 微积分基础

- 导数、梯度、泰勒展开

## ➤ 概率与统计基础

- 概率公式、常见分布、统计量



## ■ 基本概念

- 标量 (scalar) : 一个标量就是一个单独的数
- 向量 (vector) : 一个向量是一列数
- 矩阵 (matrix) : 矩阵是一个二维数组, 其中的每一个元素被两个索引 (而非一个) 所确定
- 张量 (tensor) : 在某些情况下, 需要使用坐标超过两维的数组。

## ■ 矩阵运算

- 转置
- 矩阵加法
- 矩阵乘法
- 逆矩阵运算

$$[AB]_{mn} = \sum_{k=1}^K a_{mk} b_{kn}$$

Diagram illustrating matrix multiplication  $C = AB$ .

Matrix  $A$  (rows  $i$  and columns  $n$ ) is multiplied by Matrix  $B$  (columns  $j$  and rows  $p$ ) to produce Matrix  $C$ .

The element  $c_{ij}$  is calculated as the dot product of row  $i$  of  $A$  and column  $j$  of  $B$ :

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}$$



## ■ 范数

**$\ell_1$  范数**  $\ell_1$  范数为向量的各个元素的绝对值之和.

$$\|\mathbf{v}\|_1 = \sum_{n=1}^N |v_n|.$$

**$\ell_2$  范数**  $\ell_2$  范数为向量的各个元素的平方和再开平方.

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{n=1}^N v_n^2} = \sqrt{\mathbf{v}^\top \mathbf{v}}.$$

**$\ell_\infty$  范数**  $\ell_\infty$  范数为向量的各个元素的最大绝对值,

$$\|\mathbf{v}\|_\infty = \max\{v_1, v_2, \dots, v_N\}.$$

## ■ 矩阵的范数, 常用的 $\ell_p$ 范数定义

矩阵的范数有很多种形式, 其中常用的  $\ell_p$  范数定义为

$$\|\mathbf{A}\|_p = \left( \sum_{m=1}^M \sum_{n=1}^N |a_{mn}|^p \right)^{1/p}.$$

## ■ 矩阵的F范数是向量的 $\ell_2$ 范围的推广

$$\|\mathbf{W}\|_F = \sqrt{\sum_{m=1}^M \sum_{n=1}^N (w_{mn})^2}$$



■ 导数

- 曲线的斜率，反应曲线变化的快慢

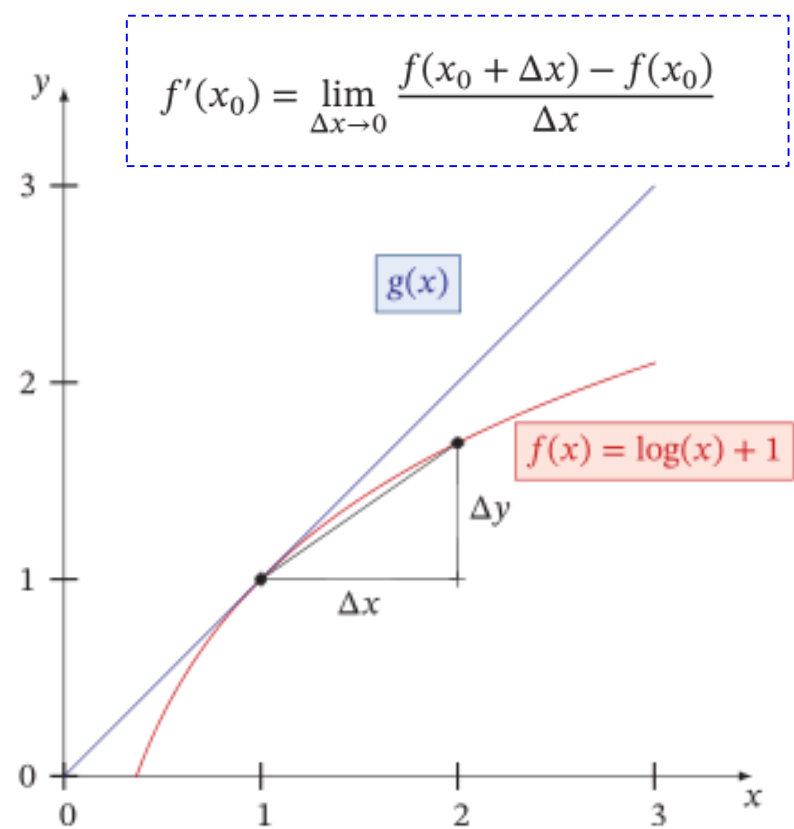


图 B.1 函数  $f(x) = \log(x) + 1$  的导数

■ 常见函数的导数

函数	函数形式	导数
常函数	$f(x) = C$ , 其中 $C$ 为常数	$f'(x) = 0$
幂函数	$f(x) = x^r$ , 其中 $r$ 是非零实数	$f'(x) = rx^{r-1}$
指数函数	$f(x) = \exp(x)$	$f'(x) = \exp(x)$
对数函数	$f(x) = \log(x)$	$f'(x) = \frac{1}{x}$

■ 高阶导数、偏导数

- 高阶导数：导数的继续求导
- 偏导数：关于其中一个变量的导数， 而保持其他变量固定



## ■ 泰勒公式

- 一个函数  $f(x)$  在已知某一点的各阶导数值的情况之下，可以用这些导数值做系数构建一个多项式来近似函数在这一点邻域中的值。

$$f(x) = f(a) + \frac{1}{1!}f'(a)(x-a) + \frac{1}{2!}f^{(2)}(a)(x-a)^2 + \dots \\ + \frac{1}{n!}f^{(n)}(a)(x-a)^n + \underline{R_n(x)},$$

泰勒公式的余项，  
 $(x-a)^n$ 的高阶无穷小

## ■ 导数与梯度

- 梯度的方向是函数在该点变化最快的方向

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta},$$



## ■ 矩阵微分

✓ 多元微积分的一种表达方式，即使用矩阵和向量来表示因变量每个成分关于自变量每个成分的偏导数。

✓ 分母布局

标量关于向量的偏导数

$$\frac{\partial y}{\partial \mathbf{x}} = \left[ \frac{\partial y}{\partial x_1}, \dots, \frac{\partial y}{\partial x_p} \right]^T$$

向量关于向量的偏导数

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_q}{\partial x_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial y_1}{\partial x_p} & \dots & \frac{\partial y_q}{\partial x_p} \end{bmatrix} \in \mathbb{R}^{p \times q}$$



## ■ 标量的微分链式法则

若  $x \in \mathbb{R}$ ,  $y = g(x) \in \mathbb{R}$ ,  $z = f(y) \in \mathbb{R}$ , 则链式法则为:

$$\frac{dz}{dx} = \frac{dy}{dx} \frac{dz}{dy}$$

## ■ 向量的微分链式法则

✓ 若  $x \in \mathbb{R}$ ,  $\mathbf{y} = g(x) \in \mathbb{R}^M$ ,  $\mathbf{z} = f(\mathbf{y}) \in \mathbb{R}^N$ , 则链式法则为:

$$\frac{\partial \mathbf{z}}{\partial x} = \frac{\partial \mathbf{y}}{\partial x} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \in \mathbb{R}^{1 \times M} \mathbb{R}^{M \times N} = \mathbb{R}^{1 \times N}$$

✓ 若  $\mathbf{x} \in \mathbb{R}^M$ ,  $\mathbf{y} = g(\mathbf{x}) \in \mathbb{R}^N$ ,  $\mathbf{z} = f(\mathbf{y}) \in \mathbb{R}^K$ , 则链式法则为:

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \in \mathbb{R}^{M \times N} \mathbb{R}^{N \times K} = \mathbb{R}^{M \times K}$$

✓ 若  $\mathbf{X} \in \mathbb{R}^{M \times N}$ ,  $\mathbf{y} = g(\mathbf{X}) \in \mathbb{R}^N$ ,  $z = f(\mathbf{y}) \in \mathbb{R}$ , 则链式法则为:

$$\frac{\partial z}{\partial x_{ij}} = \frac{\partial \mathbf{y}}{\partial x_{ij}} \frac{\partial z}{\partial \mathbf{y}} \in \mathbb{R}^{1 \times N} \mathbb{R}^{N \times 1} = \mathbb{R}$$



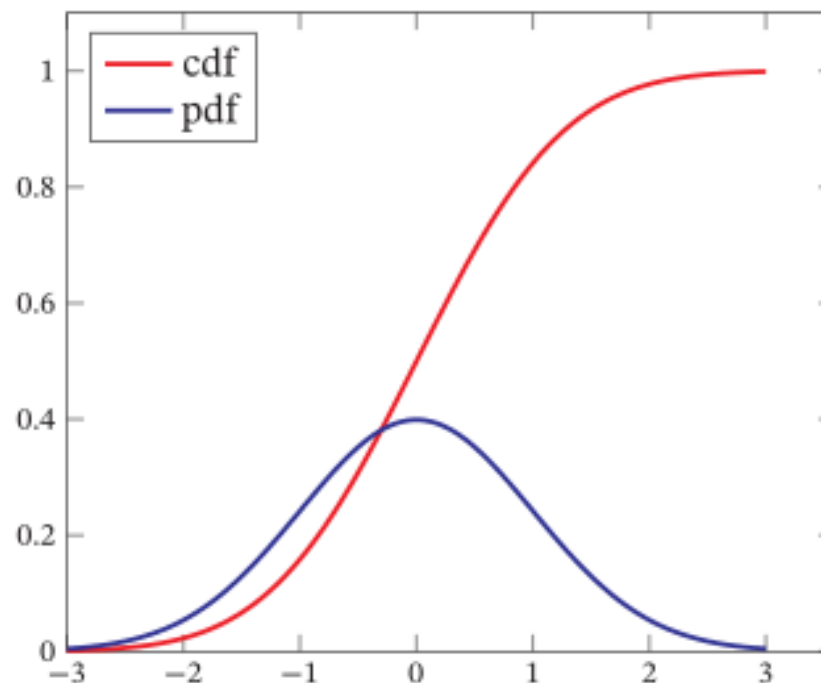


## ■ 概率公式

- 条件概率  $p(y|x) \triangleq P(Y = y|X = x) = \frac{p(x, y)}{p(x)}.$
- 贝叶斯公式  $p(y|x) = \frac{p(x|y)p(y)}{p(x)}.$

## ■ 常见概率分布

- 离散随机变量的概率分布有：
  - 伯努利分布、二项分布
- 连续随机变量的概率分布有：
  - 均匀分布、正态分布



标准正态分布的概率密度函数和累计分布函数



## ■ Jensen不等式

如果  $X$  是随机变量,  $g$  是凸函数, 则

$$g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)]$$

等式当且仅当  $X$  是一个常数或  $g$  是线性时成立

## ■ 大数定理

## ■ 随机过程

## ■ 信息熵

...

统计学习 <sup>?</sup> == 机器学习



## 2.1 机器学习基本概念

## 2.2 数学基础

## 2.3 线性模型

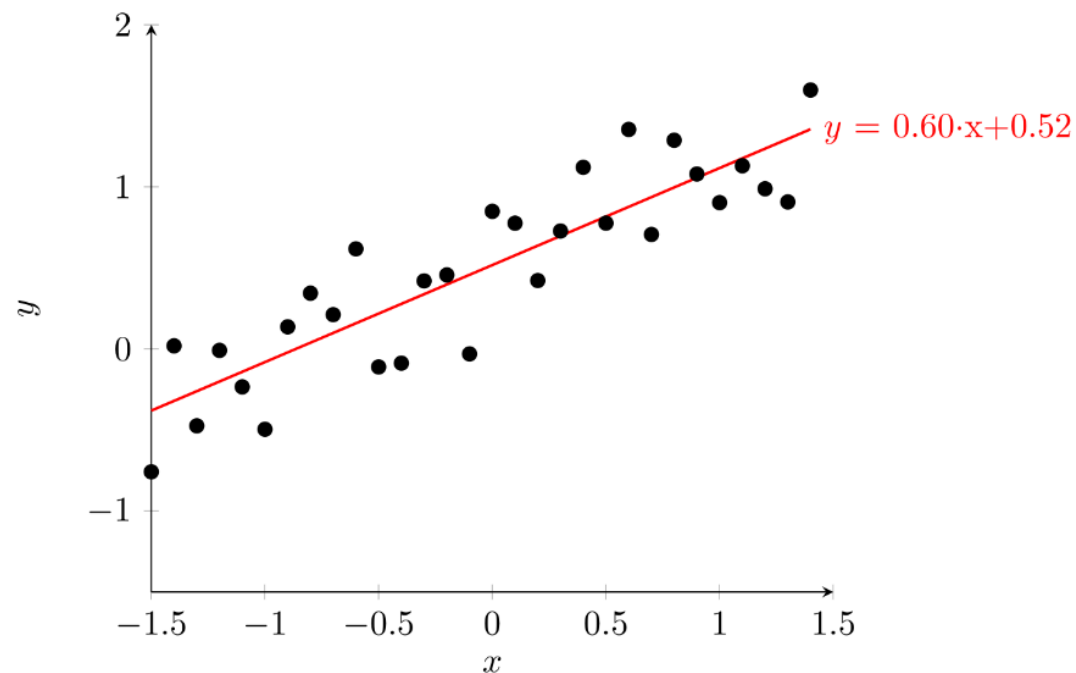
- Logistic回归
- Softmax回归
- 感知机



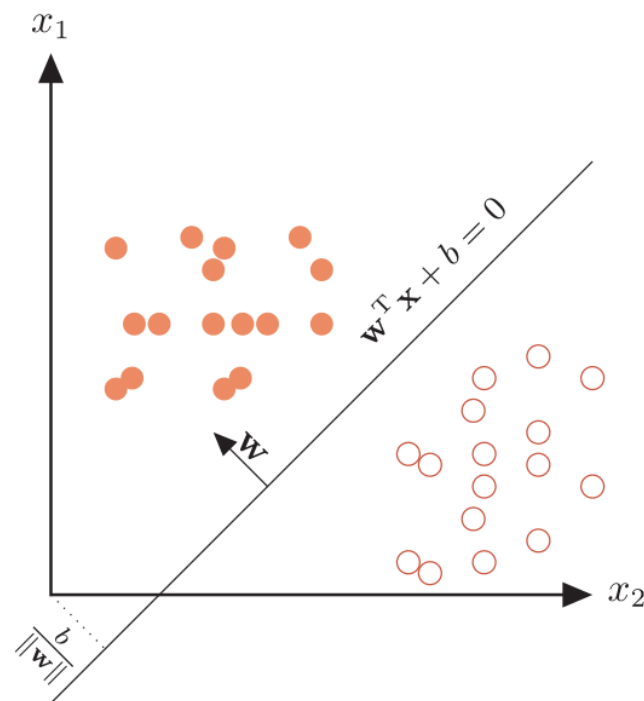
# 线性模型 (Linear Model)

- 线性回归：通过样本特征的线性组合来进行连续值的预测

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T \mathbf{x} + b$$



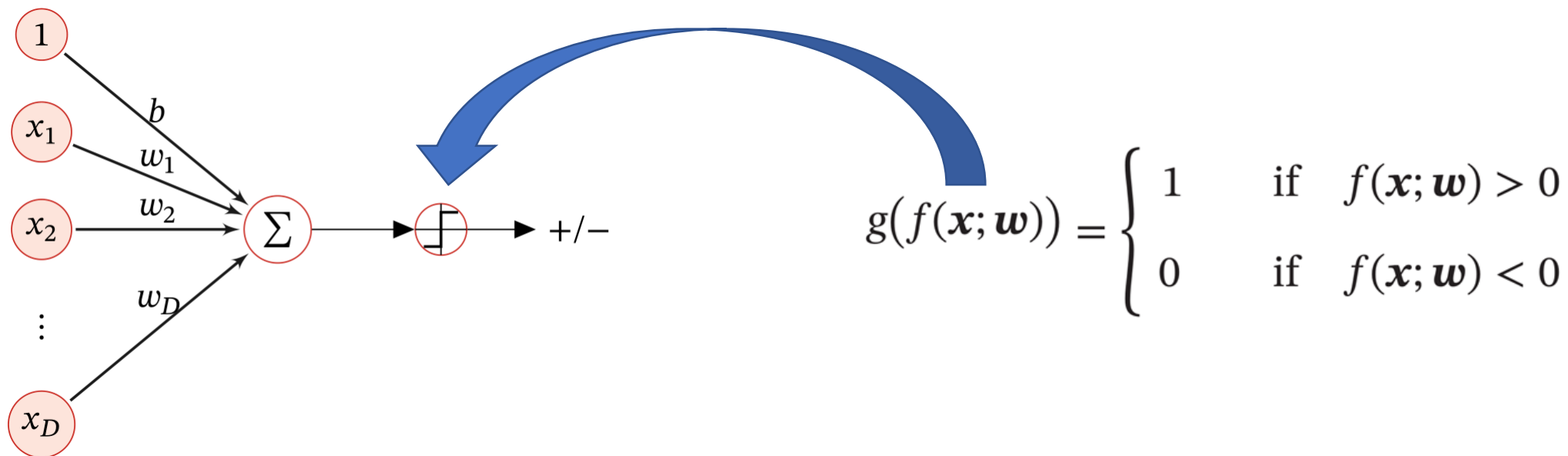
- 分类问题中，由于输出目标  $y$  是一些离散的标签，而  $f$  值域为实数，因此无法直接进行预测



线性判别函数 + 非线性决策函数



- 例如，在二分类问题中，决策函数  $g(\cdot)$  可以是符号函数(Sign Function)



是否还有其他的决策函数？



## ■ 将分类决策问题看作条件概率估计问题

- 函数  $f$  为线性函数  $f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T \mathbf{x} + b$
- 引入非线性函数  $g$  来计算类别标签的条件概率  $p(y = c | \mathbf{x})$ 。

$$p(y = 1 | \mathbf{x}) = g(f(\mathbf{x}; \mathbf{w}))$$

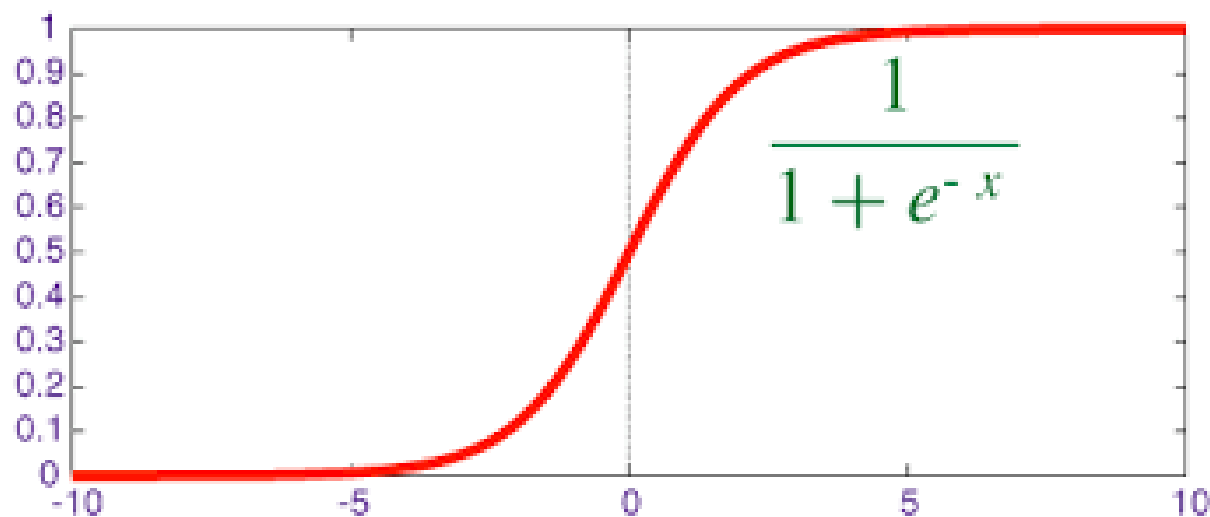
- 函数  $g$ : 把线性函数的值域从实数区间 “挤压” 到了  $(0,1)$  之间, 可以用来表示概率。

如何构造函数  $g$  ?



## ■ Logistic函数

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



## ■ Logistic回归

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$$
$$\triangleq \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$



## ■ 模型预测条件概率 $p_{\theta}(y|\mathbf{x})$

$$p_{\theta}(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

## ■ 真实条件概率 $p_r(y|\mathbf{x})$

- 对于一个样本  $(\mathbf{x}, y^*)$ , 其真实条件概率为

$$p_r(y = 1|\mathbf{x}) = y^*$$

$$p_r(y = 0|\mathbf{x}) = 1 - y^*$$

如何衡量两个条件概率的差异?





## ■ 熵 (Entropy)

✓ 在信息论中，熵用来衡量一个随机事件的不确定性。

- 自信息 (Self Information) :  $I(x) = -\log(p(x))$

- 熵 
$$\begin{aligned} H(X) &= \mathbb{E}_X[I(x)] \\ &= \mathbb{E}_X[-\log p(x)] \\ &= - \sum_{x \in \mathcal{X}} p(x) \log p(x) \end{aligned}$$

- 熵越高，则随机变量的信息越多；熵越低，则随机变量的信息越少。

- 在对分布  $p(y)$  的符号进行编码时，熵  $H(p)$  也是理论上最优的平均编码长度，这种编码方式称为熵编码 (Entropy Encoding)。



## ■ 交叉熵 (Cross Entropy)

- ✓ 交叉熵是按照概率分布  $q$  的最优编码对真实分布为  $p$  的信息进行编码的长度。

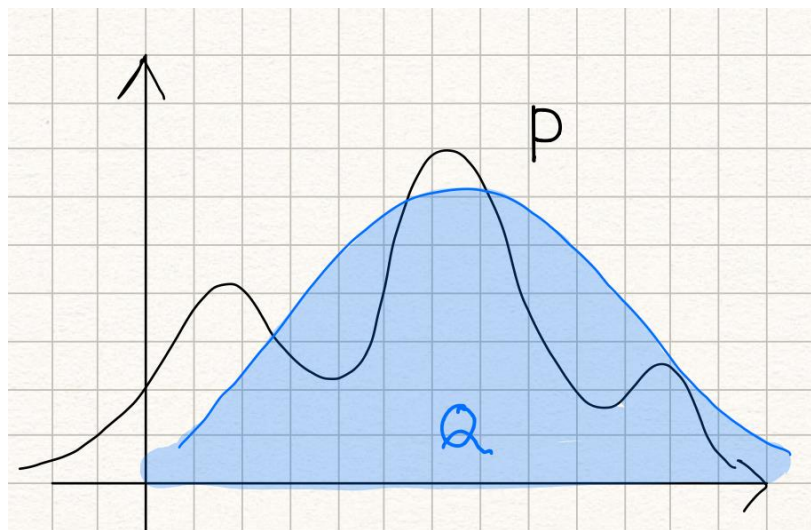
$$\begin{aligned} H(p, q) &= \mathbb{E}_p[-\log q(x)] \\ &= -\sum_x p(x) \log q(x) \end{aligned}$$

- ✓ 在给定  $q$  的情况下，如果  $p$  和  $q$  越接近，交叉熵越小。
- ✓ 如果  $p$  和  $q$  差别越大，交叉熵就越大。



## ■ KL散度 (Kullback-Leibler Divergence)

- KL散度是用概率分布  $q$  来近似  $p$  时所造成的信息损失量。



$$\begin{aligned} \text{KL}(p, q) &= H(p, q) - H(p) \\ &= \sum_x p(x) \log \frac{p(x)}{q(x)} \end{aligned}$$

- KL散度是按照概率分布  $q$  的最优编码对真实分布为  $p$  的信息进行编码，其平均编码长度（即交叉熵） $H(p, q)$  和  $p$  的最优平均编码长度（即熵） $H(p)$  之间的差异。



## ■ 交叉熵损失函数

$$D_{\text{KL}}(p_r(y|\mathbf{x})||p_\theta(y|\mathbf{x})) = \sum_{y=0}^1 p_r(y|\mathbf{x}) \log \frac{p_r(y|\mathbf{x})}{p_\theta(y|\mathbf{x})} \quad \text{KL散度}$$

$$\propto - \sum_{y=0}^1 p_r(y|\mathbf{x}) \log p_\theta(y|\mathbf{x}) \quad \text{交叉熵}$$

$$= -I(y^* = 1) \log p_\theta(y = 1|\mathbf{x}) - I(y^* = 0) \log p_\theta(y = 0|\mathbf{x}) \quad y^* \text{为}\mathbf{x}\text{的真实标签}$$

$$= -y^* \log p_\theta(y = 1|\mathbf{x}) - (1 - y^*) \log p_\theta(y = 0|\mathbf{x}) \quad \text{交叉熵损失函数}$$

$$= -\log p_\theta(y^*|\mathbf{x}) \quad \text{负对数似然}$$



## ■ 梯度下降

- 基于交叉熵损失函数，模型在训练集的风险函数为：

$$\begin{aligned}\mathcal{R}(\mathbf{w}) &= -\frac{1}{N} \sum_{i=1}^N \left( y^{(i)} \log \left( \sigma(\mathbf{w}^T \mathbf{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left( 1 - \sigma(\mathbf{w}^T \mathbf{x}^{(i)}) \right) \right) . \\ &= -\frac{1}{N} \sum_{n=1}^N \left( y^{(n)} \log \hat{y}^{(n)} + (1 - y^{(n)}) \log(1 - \hat{y}^{(n)}) \right) .\end{aligned}$$

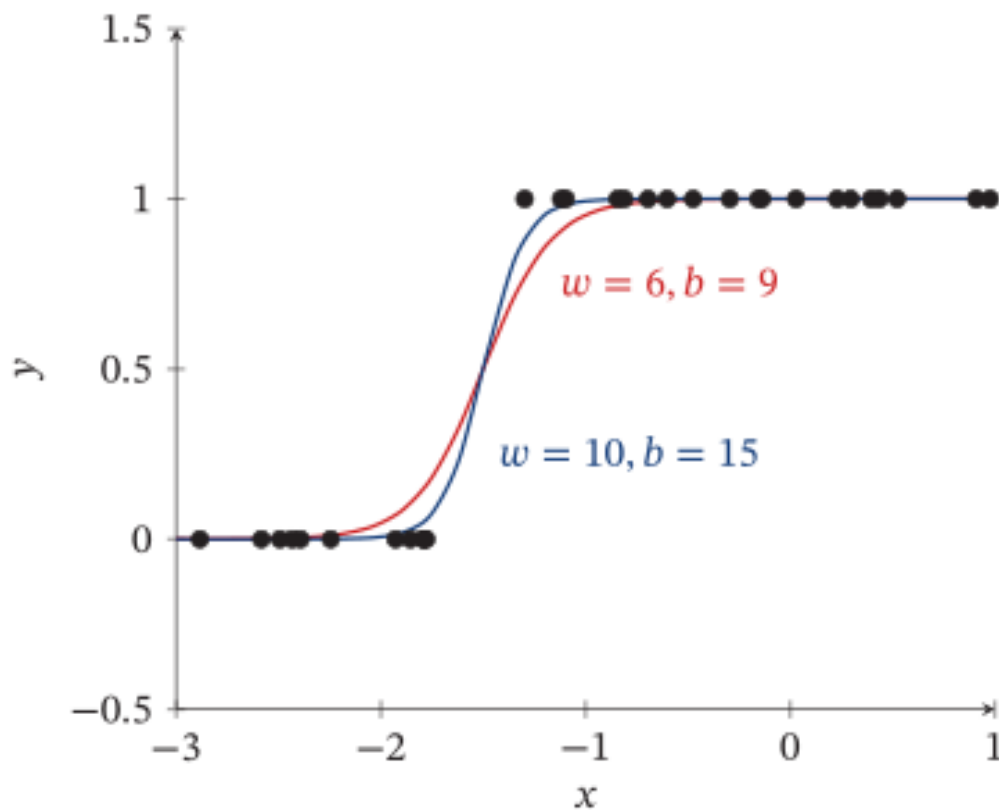
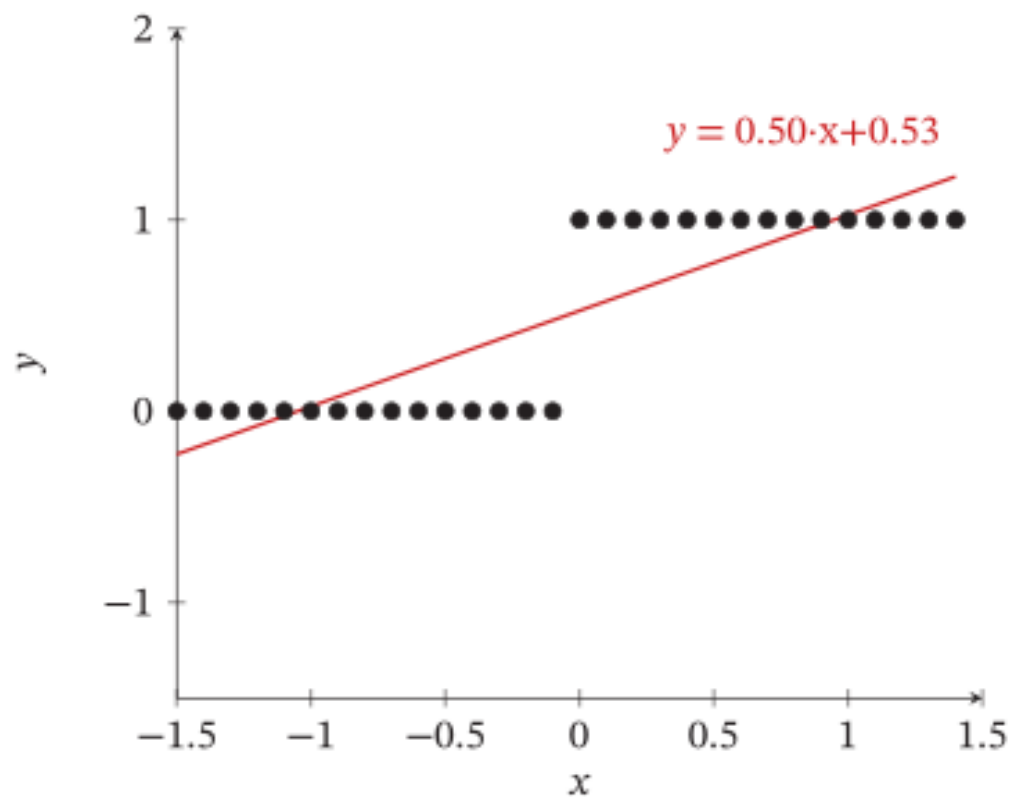
- 梯度为

$$\begin{aligned}\frac{\partial \mathcal{R}(\mathbf{w})}{\partial \mathbf{w}} &= -\frac{1}{N} \sum_{n=1}^N \left( y^{(n)} \frac{\hat{y}^{(n)}(1 - \hat{y}^{(n)})}{\hat{y}^{(n)}} \mathbf{x}^{(n)} - (1 - y^{(n)}) \frac{\hat{y}^{(n)}(1 - \hat{y}^{(n)})}{1 - \hat{y}^{(n)}} \mathbf{x}^{(n)} \right) \\ &= -\frac{1}{N} \sum_{n=1}^N \left( y^{(n)}(1 - \hat{y}^{(n)}) \mathbf{x}^{(n)} - (1 - y^{(n)}) \hat{y}^{(n)} \mathbf{x}^{(n)} \right) \\ &= -\frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)} (y^{(n)} - \hat{y}^{(n)}) .\end{aligned}$$



# Logistic回归

- 使用线性回归和Logistic回归来解决一维数据的二分类问题的示例。

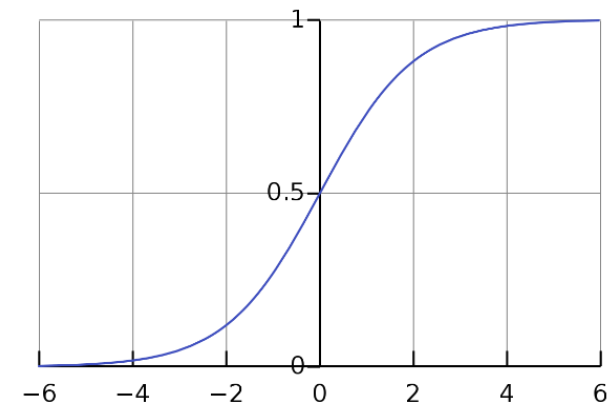




# Logistic回归-小结

## ➤ 模型

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) \triangleq \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$



## ➤ 学习准则：交叉熵损失

$$\mathcal{R}(\mathbf{w}) = -\frac{1}{N} \sum_{n=1}^N \left( y^{(n)} \log \hat{y}^{(n)} + (1 - y^{(n)}) \log(1 - \hat{y}^{(n)}) \right)$$

## ➤ 优化算法：梯度下降

$$\frac{\partial \mathcal{R}(\mathbf{w})}{\partial \mathbf{w}} = -\frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)} (y^{(n)} - \hat{y}^{(n)})$$



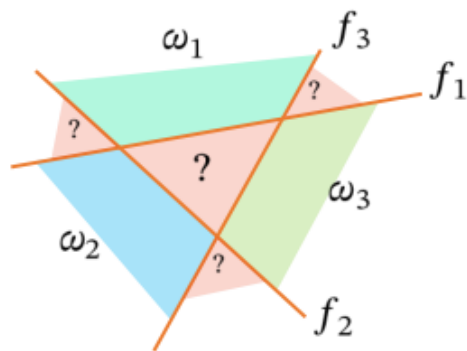
# 多分类问题 (Multi-class Classification)

- 多分类问题是指分类的类别数大于2。多分类一般需要多个线性判别函数，设计这些判别函数有很多多种方式。

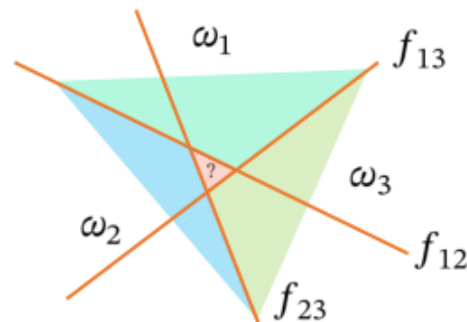
假设一个多分类问题的类别为 $\{1, 2, \dots, C\}$ ，常用的方式有以下三种：

- “一对其余” 方式：把多分类问题转换为  $C$  个 “一对其余” 的二分类问题。这种方式共需要  $C$  个判别函数，其中第  $c$  个判别函数  $f_c$  是将类别  $c$  的样本和不属于类别  $c$  的样本分开。
- “一对一” 方式：把多分类问题转换为  $C(C-1)/2$  个 “一对一” 的二分类问题。这种方式共需要  $C(C-1)/2$  个判别函数，其中第  $(i, j)$  个判别函数是把类别  $i$  和类别  $j$  的样本分开。
- “argmax” 方式：这是一种改进的 “一对其余” 方式，共需要  $C$  个判别函数

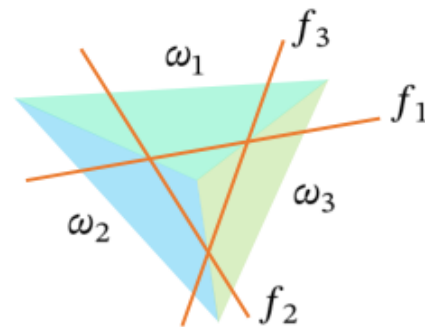
$$f_c(\mathbf{x}; \mathbf{w}_c) = \mathbf{w}_c^T \mathbf{x} + b_c,$$
$$c \in \{1, \dots, C\}$$



(a) “一对其余”方式



(b) “一对一”方式



(c) “argmax”方式





## ■ 多分类问题的预测函数

$$y = \arg \max_{c=1}^C f_c(\mathbf{x}; \mathbf{w}_c)$$

## ■ Softmax函数

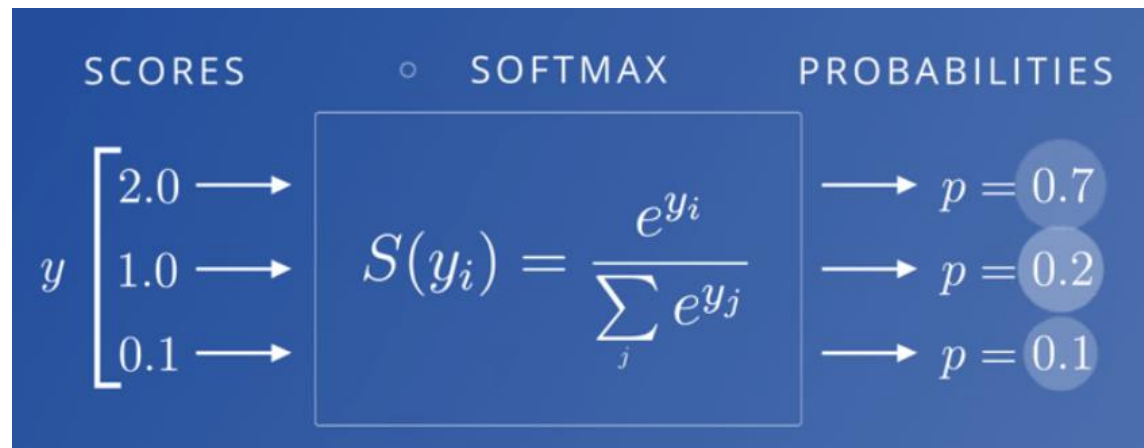
$$\text{softmax}(x_k) = \frac{\exp(x_k)}{\sum_{i=1}^K \exp(x_i)}$$





- 利用softmax函数，目标类别 $y = c$ 的条件概率为：

$$P(y = c|\mathbf{x}) = \text{softmax}(\mathbf{w}_c^T \mathbf{x}) \\ = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{i=1}^C \exp(\mathbf{w}_i^T \mathbf{x})}.$$



- Softmax回归也称为多项或多类的Logistic回归



## ■ 交叉熵损失

- KL散度

$$D_{\text{KL}}(p_r(y|\mathbf{x})||p_\theta(y|\mathbf{x})) = \sum_{y=1}^c p_r(y|\mathbf{x}) \log \frac{p_r(y|\mathbf{x})}{p_\theta(y|\mathbf{x})}$$

$$\propto - \sum_{y=1}^c p_r(y|\mathbf{x}) \log p_\theta(y|\mathbf{x}) \quad \text{交叉熵损失}$$

$$= -\log p_\theta(y^*|\mathbf{x}) \quad \text{负对数似然}$$

$y^*$ 为 $x$ 的真实标签



## ■ 交叉熵损失

- 风险函数

$$\begin{aligned}\mathcal{R}(W) &= -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C y_c^{(n)} \log \hat{y}_c^{(n)} \\ &= -\frac{1}{N} \sum_{n=1}^N (\mathbf{y}^{(n)})^\top \log \hat{\mathbf{y}}^{(n)},\end{aligned}$$

- 风险函数的梯度为

$$\frac{\partial \mathcal{R}(W)}{\partial W} = -\frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)} \left( \mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)} \right)^\top$$



## ➤模型

$$\begin{aligned} P(y = c|\mathbf{x}) &= \text{softmax}(\mathbf{w}_c^\top \mathbf{x}) \\ &= \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_{i=1}^C \exp(\mathbf{w}_i^\top \mathbf{x})}. \end{aligned}$$

## ➤学习准则：交叉熵损失

$$\mathcal{R}(W) = -\frac{1}{N} \sum_{n=1}^N (\mathbf{y}^{(n)})^\top \log \hat{\mathbf{y}}^{(n)}$$

## ➤优化算法：梯度下降

$$\frac{\partial \mathcal{R}(W)}{\partial W} = -\frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)} \left( \mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)} \right)^\top$$



# 感知机 (Perceptron)

- 感知机由 Frank Rosenblatt 于1958年提出，是一种广泛使用的线性分类器。感知器可谓是最简单的人工神经网络，只有一个神经元。

*Psychological Review*  
Vol. 65, No. 6, 1958

## THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN<sup>1</sup>

F. ROSENBLATT  
*Cornell Aeronautical Laboratory*

HAVING told you about the giant digital computer known as **L.B.M. 704** and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

*The New Yorker*, December 6, 1958 P. 44



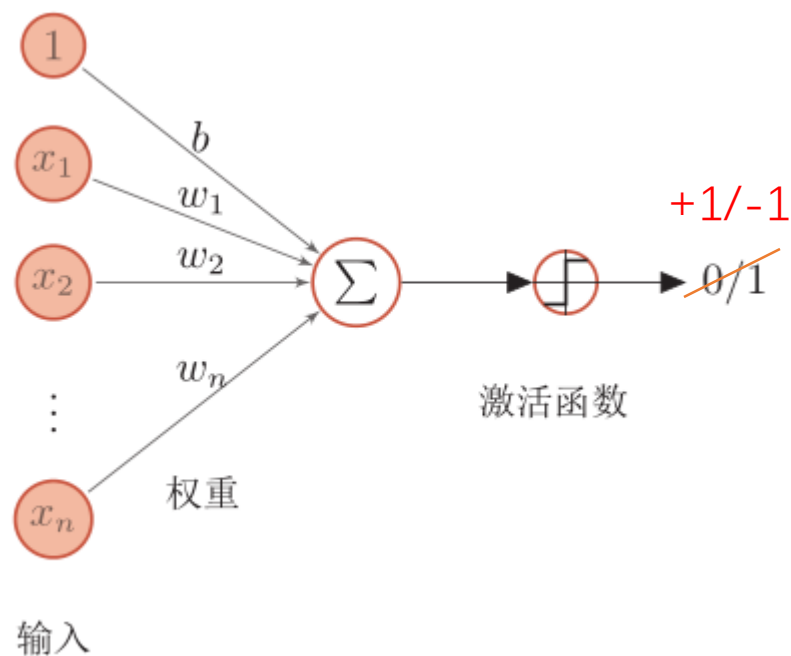
The IBM 704 computer



# 感知机 (Perceptron)

- 模拟生物神经元行为的机器，有与生物神经元相对应的部件，如权重（突触）、偏置（阈值）及激活函数（细胞体），输出为+1或-1。

$$\hat{y} = \begin{cases} +1 & \text{当 } \mathbf{w}^T \mathbf{x} > 0 \\ -1 & \text{当 } \mathbf{w}^T \mathbf{x} \leq 0 \end{cases},$$





## ➤ 学习算法

- 一种错误驱动的在线学习算法：
- 先初始化一个权重向量  $\mathbf{w} \leftarrow \mathbf{0}$  (通常是全零向量) ；
- 每次分错一个样本  $(\mathbf{x}, y)$  时， 即

$$y\mathbf{w}^T \mathbf{x} < 0$$

- 用这个样本来更新权重

$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$$

- 根据感知器的学习策略， 可以反推出感知器的损失函数为

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, y) = \max(0, -y\mathbf{w}^T \mathbf{x})$$





## ■ 感知器的学习过程


---

### 算法 3.1: 两类感知器的参数学习算法

---

输入: 训练集  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ , 最大迭代次数  $T$

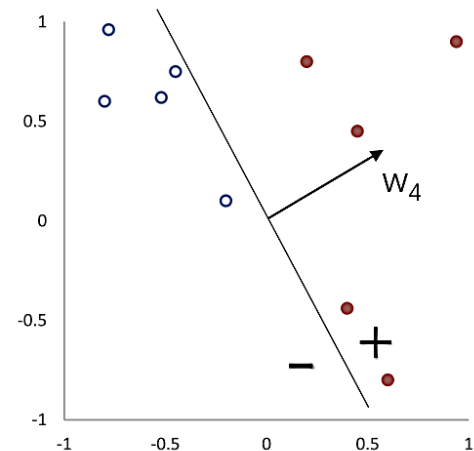
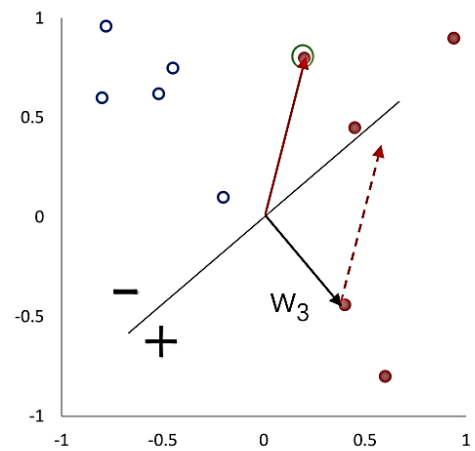
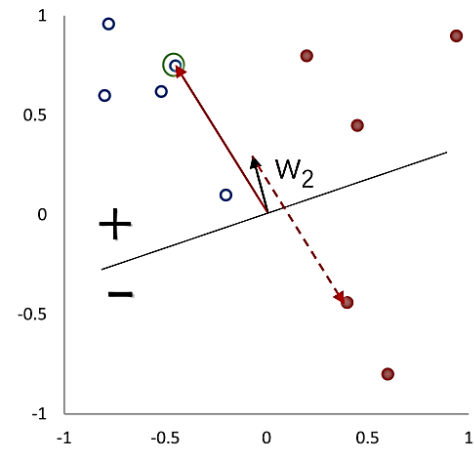
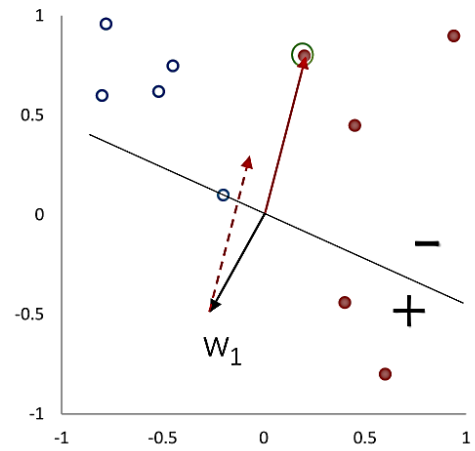
```
1 初始化:  $\mathbf{w}_0 \leftarrow 0, k \leftarrow 0, t \leftarrow 0$ ;  
2 repeat  
3   对训练集  $\mathcal{D}$  中的样本随机排序;  
4   for  $n = 1 \cdots N$  do  
5     选取一个样本  $(\mathbf{x}^{(n)}, y^{(n)})$ ;  
6     if  $\mathbf{w}_k^\top (y^{(n)} \mathbf{x}^{(n)}) \leq 0$  then // 表示分错  
7        $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + y^{(n)} \mathbf{x}^{(n)}$ ;  
8        $k \leftarrow k + 1$ ;  
9     end  
10     $t \leftarrow t + 1$ ;  
11  end  
12 until  $t = T$ ;  
    输出:  $\mathbf{w}_k$ 
```





# 感知机 (Perceptron)

## ■ 感知器参数学习的更新过程





## ■ 收敛性

当数据集是两类线性可分时,对于训练集  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ , 其中  $\mathbf{x}^{(n)}$  为样本的增广特征向量,  $y^{(n)} \in \{-1, 1\}$ , 那么存在一个正的常数  $\gamma (\gamma > 0)$  和权重向量  $\mathbf{w}^*$ , 并且  $\|\mathbf{w}^*\| = 1$ , 对所有  $n$  都满足  $(\mathbf{w}^*)^\top (y^{(n)} \mathbf{x}^{(n)}) \geq \gamma$ . 我们可以证明如下定理.

**定理 3.1** – 感知器收敛性: 给定训练集  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ , 令  $R$  是训练集中最大的特征向量的模, 即

$$R = \max_n \|\mathbf{x}^{(n)}\|.$$

如果训练集  $\mathcal{D}$  线性可分, 两类感知器的参数学习算法3.1的权重更新次数不超过  $\frac{R^2}{\gamma^2}$ .



# 感知机 (Perceptron) -小结

## ➤模型

$$g(\mathbf{x}, \mathbf{w}) = \begin{cases} +1 & \text{当 } \mathbf{w}^T \mathbf{x} > 0, \\ -1 & \text{当 } \mathbf{w}^T \mathbf{x} < 0. \end{cases}$$

## ➤学习准则:

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, y) = \max(0, -y\mathbf{w}^T \mathbf{x}).$$

## ➤优化算法: 随机梯度下降

$$\frac{\partial \mathcal{L}(\mathbf{w}; \mathbf{x}, y)}{\partial \mathbf{w}} = \begin{cases} 0 & \text{当 } y\mathbf{w}^T \mathbf{x} > 0, \\ -y\mathbf{x} & \text{当 } y\mathbf{w}^T \mathbf{x} < 0. \end{cases}$$



# 线性分类模型小结

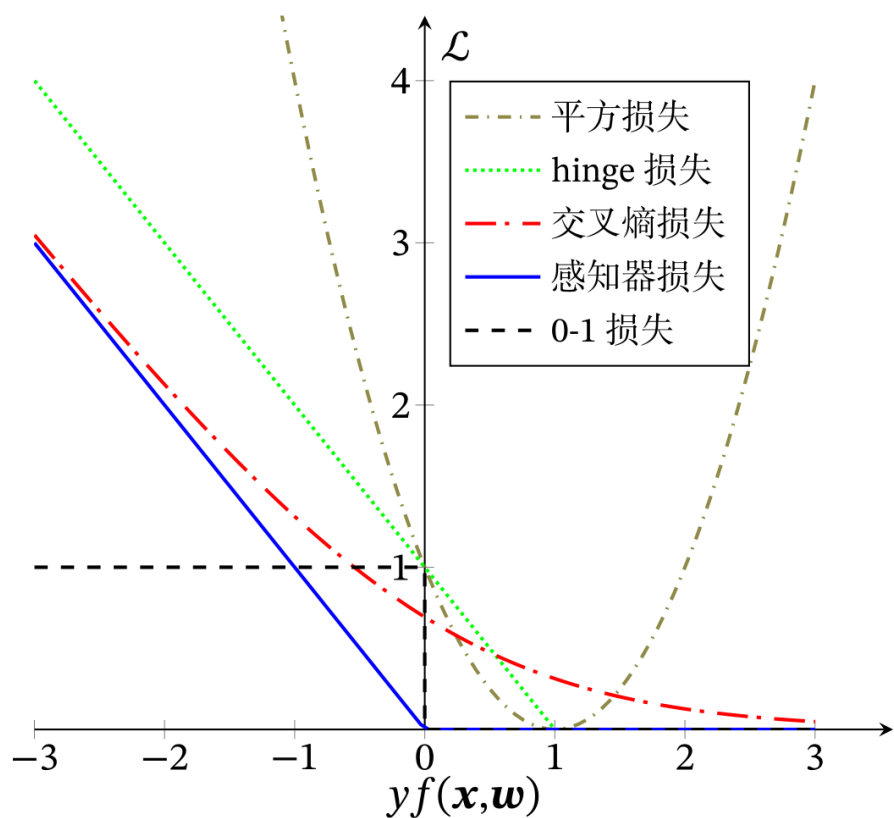
线性模型	激活函数	损失函数	优化方法
线性回归	-	$(y - \mathbf{w}^\top \mathbf{x})^2$	最小二乘、梯度下降
Logistic 回归	$\sigma(\mathbf{w}^\top \mathbf{x})$	$\mathbf{y} \log \sigma(\mathbf{w}^\top \mathbf{x})$	梯度下降
Softmax 回归	$\text{softmax}(\mathbf{W}^\top \mathbf{x})$	$\mathbf{y} \log \text{softmax}(\mathbf{W}^\top \mathbf{x})$	梯度下降
感知器	$\text{sgn}(\mathbf{w}^\top \mathbf{x})$	$\max(0, -y\mathbf{w}^\top \mathbf{x})$	随机梯度下降

- 在logistic回归和softmax回归中,  $\mathbf{y}$  为类别的one-hot向量表示;
- 在感知器中,  $y \in \{+1, -1\}$



# 不同损失函数的对比

为了比较这些损失函数，我们统一定义类别标签  $y \in \{+1, -1\}$ ，并定义  $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x} + b$ 。这样对于样本  $(\mathbf{x}, y)$ ，若  $yf(\mathbf{x}; \mathbf{w}) > 0$ ，则分类正确，相反则分类错误。这样为了方便比较这些模型，我们可以将它们的损失函数都表述为定义在  $yf(\mathbf{x}; \mathbf{w})$  上的函数。



$$\mathcal{L}_{LR} = \log (1 + \exp (-yf(\mathbf{x}; \mathbf{w})))$$

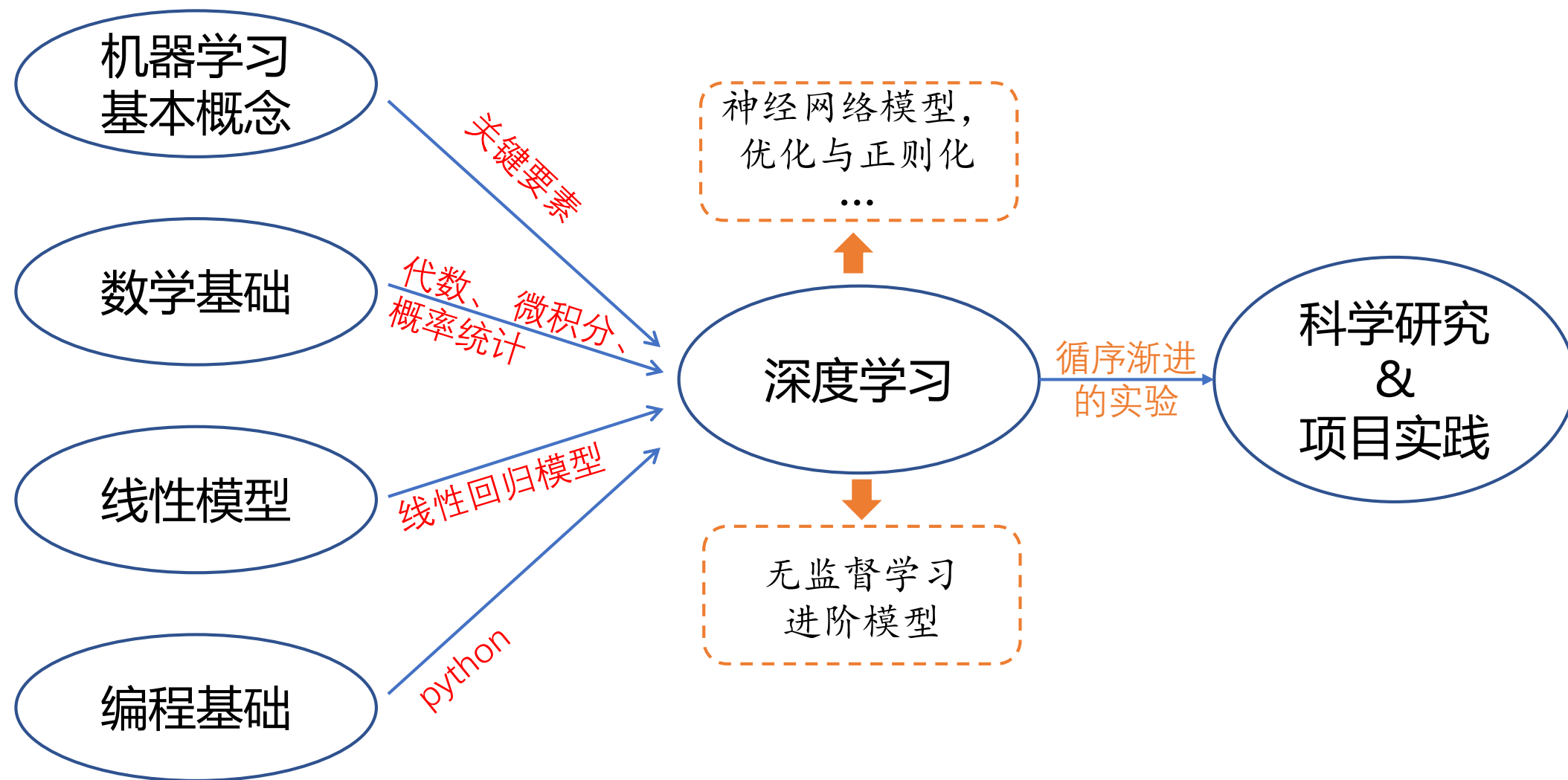
$$\mathcal{L}_{hinge} = \max (0, 1 - yf(\mathbf{x}; \mathbf{w}))$$

$$\mathcal{L}_p = \max (0, -yf(\mathbf{x}; \mathbf{w}))$$

$$\mathcal{L}_{squared} = (1 - yf(\mathbf{x}; \mathbf{w}))^2$$



# 基础知识是后续深度学习的基石







# 北京交通大学《深度学习》课程组成员

景丽萍: <http://faculty.bjtu.edu.cn/8249/>

桑基韬: <http://faculty.bjtu.edu.cn/9129/>

张淳杰: <http://faculty.bjtu.edu.cn/9371/>

万怀宇: <http://faculty.bjtu.edu.cn/8793/>

滕 竹: <http://faculty.bjtu.edu.cn/8902/>

原继东: <http://faculty.bjtu.edu.cn/9076/>

丛润民: <http://faculty.bjtu.edu.cn/9374/>

夏佳楠: <http://faculty.bjtu.edu.cn/9430/>

许万茹

杨 扩

