



南京大學
NANJING UNIVERSITY

人工智能导论

机器学习初步

郭兰哲

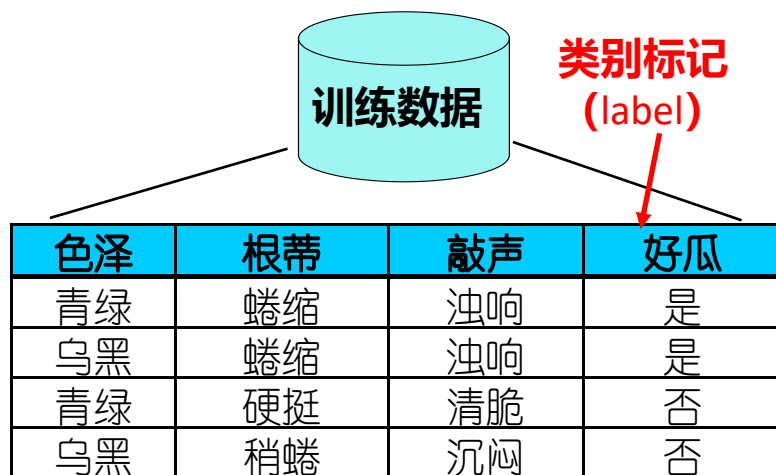
南京大学 智能科学与技术学院

<https://www.lamda.nju.edu.cn/guolz/IntroAI/fall2025/index.html>

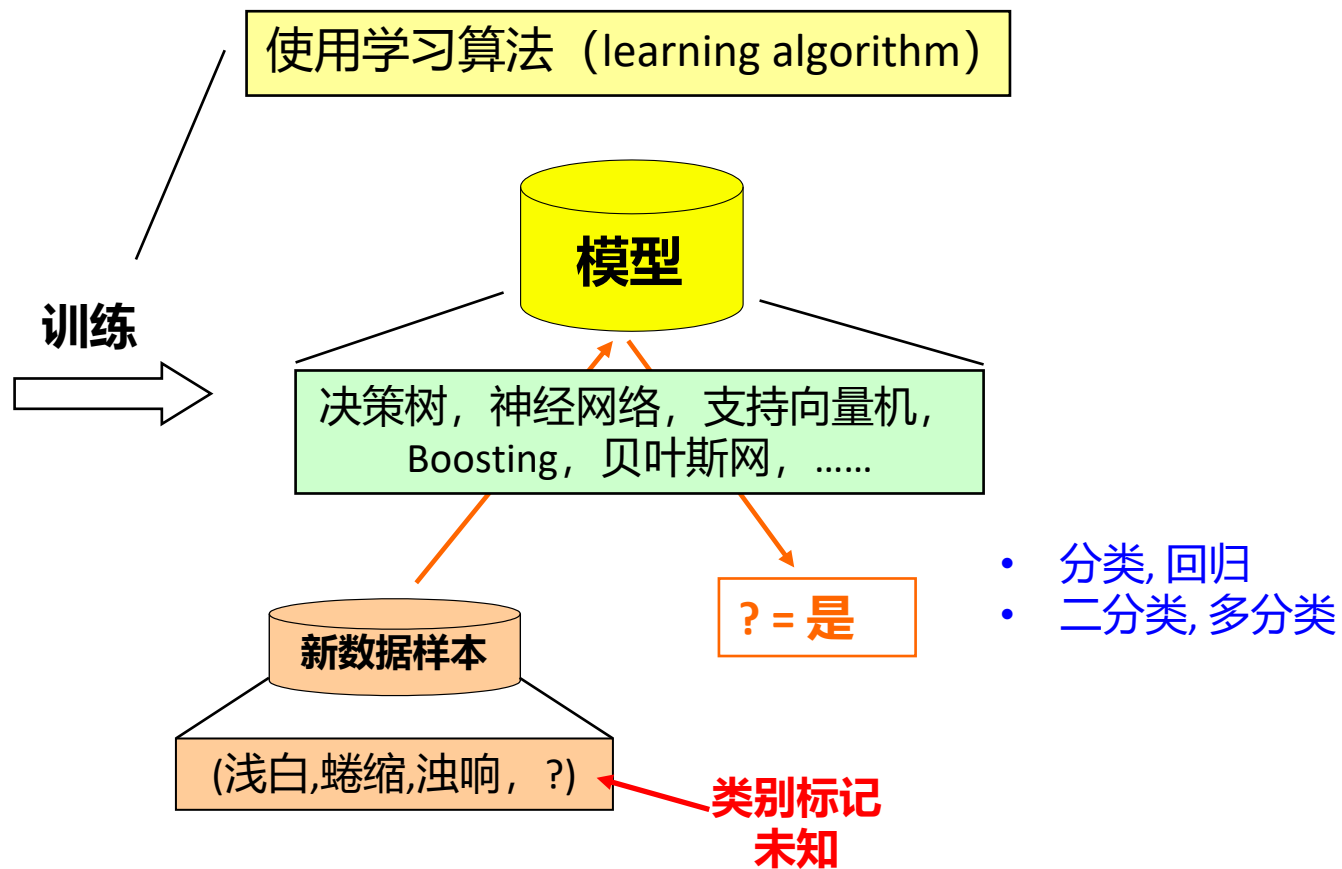
Email: guolz@nju.edu.cn

典型的机器学习过程

- 监督学习(supervised learning)
- 无监督学习(unsupervised learning)
- 强化学习(reinforcement learning)



- 数据集: 训练集、测试集
- 示例(instance), 样例(example), 样本(sample)
- 属性(attribute), 特征(feature)
- 属性值
- 属性空间, 样本空间, 输入空间
- 特征向量(feature vector)
- 标记空间, 输出空间



潜在意义

横：输入（数据） -> 输出（标记）

纵：
历史（数据） -
>
未来（数据）

		特征			标记
		↑			↑
	编号	色泽	根蒂	敲声	好瓜
训练集	1	青绿	蜷缩	浊响	是
	2	乌黑	蜷缩	沉闷	是
	3	青绿	硬挺	清脆	否
	4	乌黑	稍蜷	沉闷	否
测试集	← 1	青绿	蜷缩	沉闷	?

机器学习：面向未来的技术

任务 数据 目标 算法

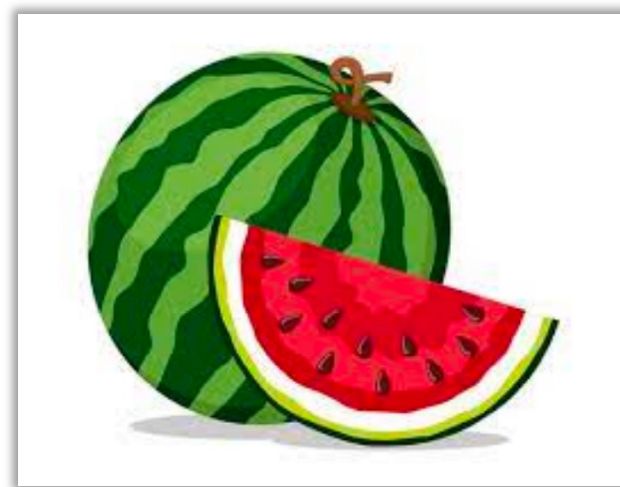
Machine Learning = task + data + objective + algorithm

--Tom Mitchell

Learning = Task + Data + Objective + Algorithm

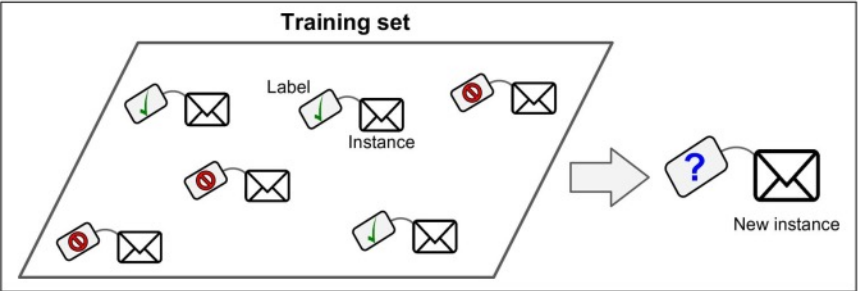
- 任务通常可以指学习一个从特征空间到类别空间的映射 $f: X \rightarrow Y$
- 以西瓜任务为例：
 - X : 西瓜的特征，如颜色、根蒂的蜷缩程度、敲击的声音等
 - Y : 西瓜是好还是坏
- f 通常来自一个约定好的空间 $\mathcal{F}$, 即 $f \in \mathcal{F}$

色泽	根蒂	敲声	好瓜
青绿	蜷缩	浊响	是
乌黑	蜷缩	浊响	是
青绿	硬挺	清脆	否
乌黑	稍蜷	沉闷	否



Learning = Task + Data + Objective + Algorithm

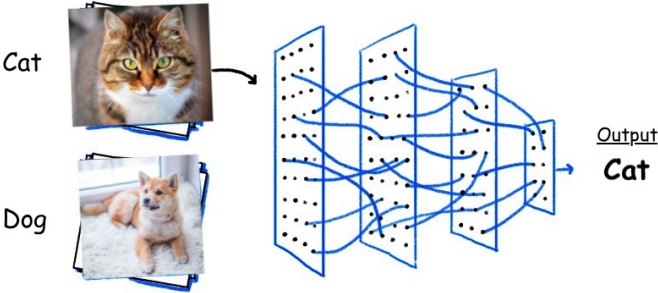
分类任务： Y 包含若干离散的属性值
二分类： $\{0, 1\}$, K 分类 $\{0, 1, \dots K\}$



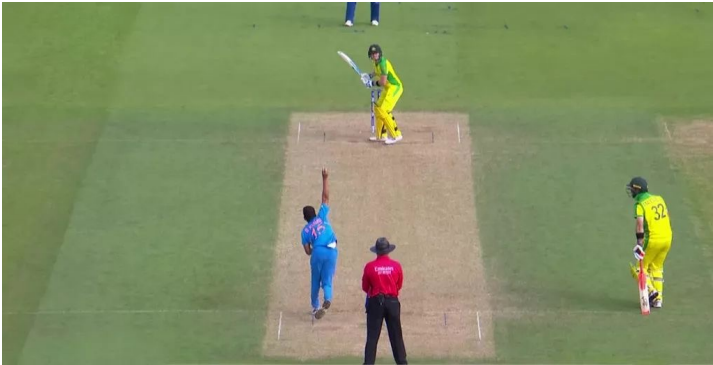
垃圾邮件分类

号码类型	主叫通话次数	主叫率	主叫外地联系人个数	主叫外地联系地个数	主叫通话频率	主叫通话时长	回拨率	联系人/通话次数比例
正常号码	少, 平均每天主叫通话6次	适中	适中, 平均每天与4人联系	极少	极低	正常, 通话时长在50s左右	正常	正常
诈骗/骚扰电话	多, 平均每天主叫通话54次	极高, 主叫比例接近100%	极多, 平均每天与51人联系	较高, 每天约与9个不同的城市联系	高, 约1h通话11次	未受骗用户识别诈骗需要一定时间, 通话时长在25s左右	几乎为0, 说明该类号码基本不会被回拨	几乎为1, 说明该类号码对多个号码进行单次拨号

欺诈检测



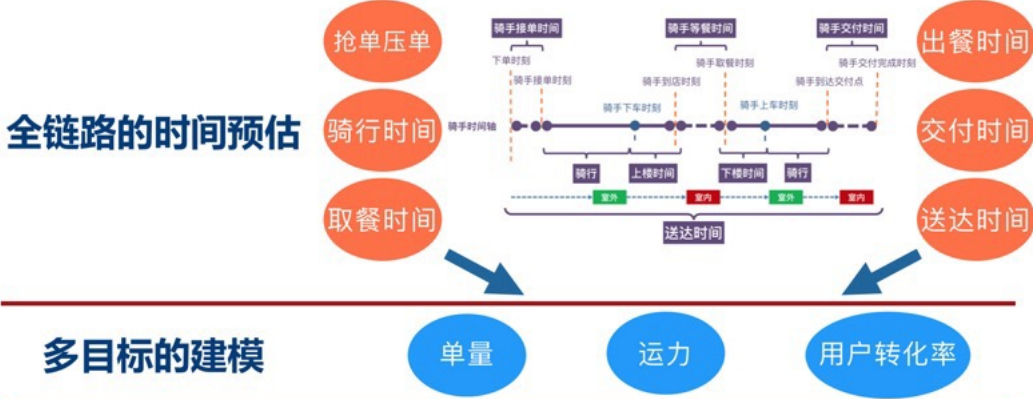
动物识别



动作分类

Learning = Task + Data + Objective + Algorithm

回归任务：Y通常是实数值



外卖送达的时间

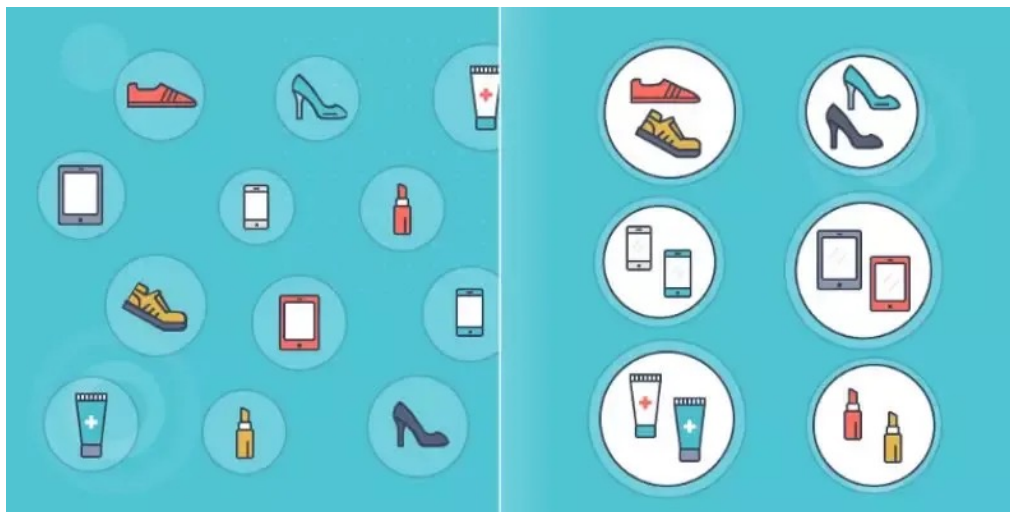


方向盘
旋转幅度
油门幅度
刹车幅度

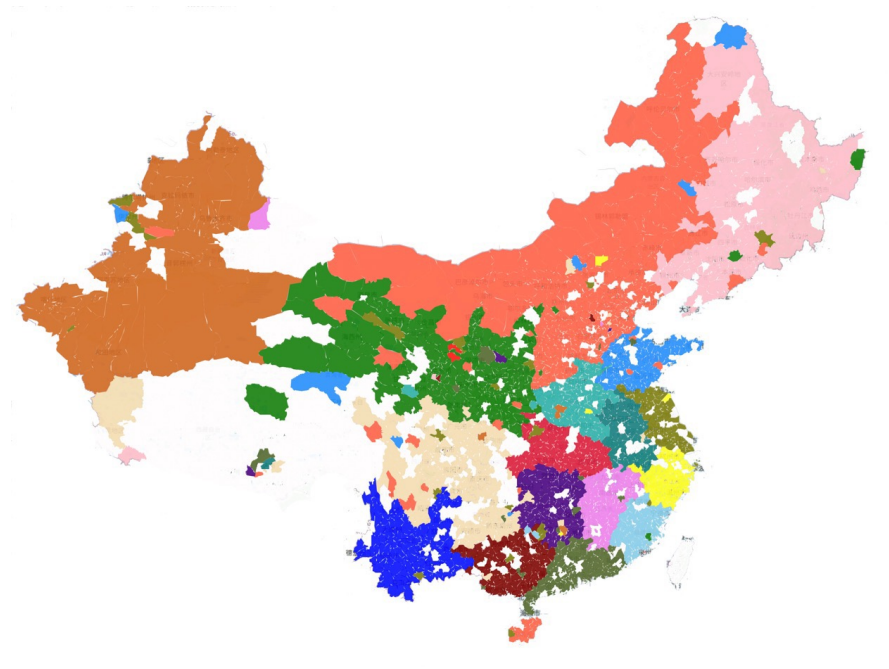
自动驾驶：方向盘旋转的幅度、油门幅度、
刹车幅度

Learning = Task + Data + Objective + Algorithm

聚类任务：把数据集中的样本划分为若干个子集



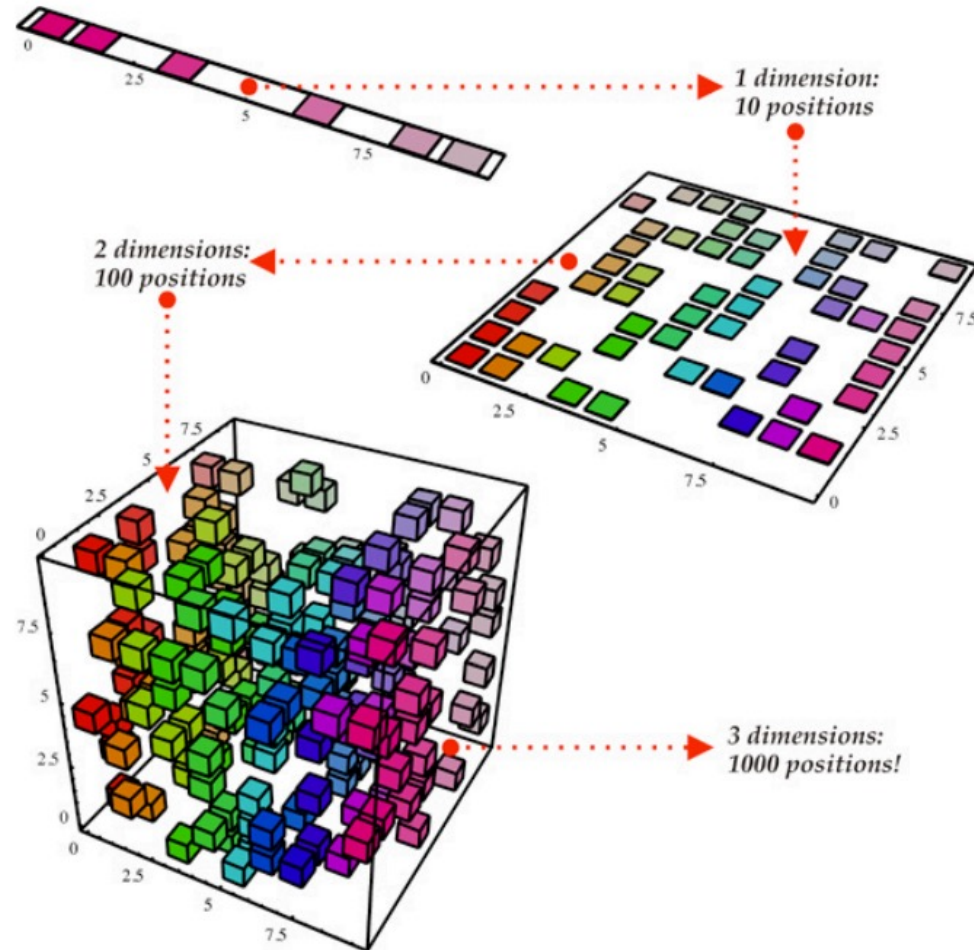
推荐系统：商品、用户聚类



区域（县）投资网络社区检测

Learning = Task + Data + Objective + Algorithm

降维任务：降低特征维度，用更少的特征表示数据

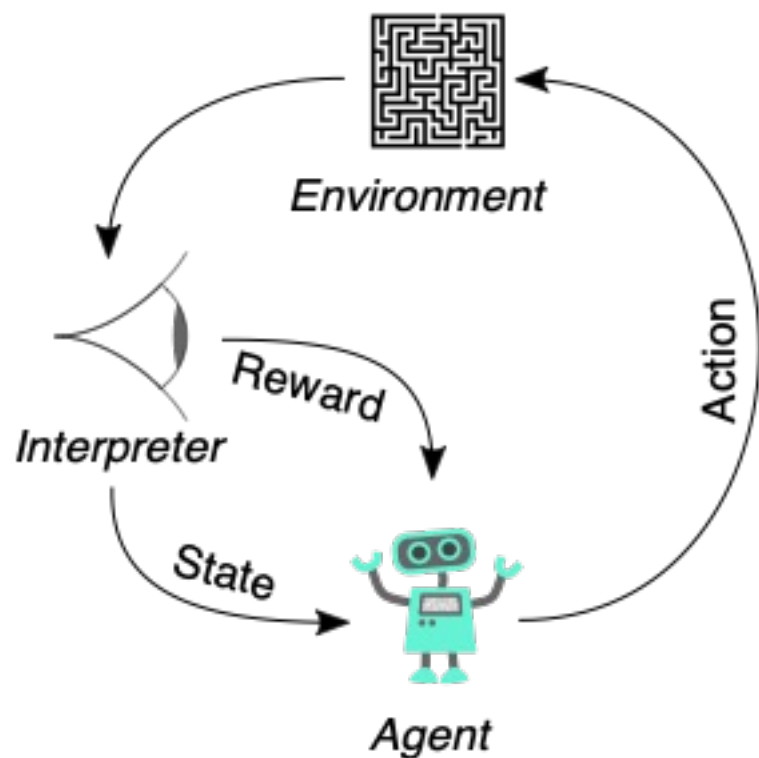


Learning = Task + **Data** + Objective + Algorithm

- **训练数据**：训练机器学习模型的基础资源
 - 监督学习： $D_{tr} = \{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\}$
 - 无监督学习： $D_{tr} = \{x_1, x_2, \dots x_n\}$
 - 半监督学习： $D_{tr} = \{(x_1, y_1), (x_2, y_2), \dots (x_t, y_t), x_{t+1}, \dots x_n\}$
- **测试数据**：评估模型的性能，在训练过程中无法见到
 - $D_{te} = \{(x_1, y_1), (x_2, y_2), \dots (x_m, y_m)\}$
- **验证数据**：用于训练过程中辅助评估模型的性能

Learning = Task + **Data** + Objective + Algorithm

强化学习：标记信息延迟的监督学习问题



Learning = Task + Data + **Objective** + Algorithm

- **评价指标/损失函数**: 反映了模型 f 的性能好坏, 用于指导模型训练

- 分类问题:

- 正确率: 模型预测正确的概率

$$P(f(x) = y) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(x_i) = y_i)$$

- 回归问题:

- 均方误差 (Mean Squared Error)

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

Learning = Task + Data + Objective + **Algorithm**

机器学习算法 $\mathcal{A}$

- 输入：训练数据集 D_{tr} , 评价指标 $M(f)$ /损失函数 $loss(f)$, f 的函数空间 $\mathcal{F}$
- 输出：学得的模型 f

$$\mathcal{A}: \mathcal{F} \times \mathcal{M} \times \mathcal{D} \rightarrow f$$

学习算法运行的过程称为模型的训练过程

即, 在所有可能的 f 组成的空间中进行**优化**的过程

经验风险最小化 (empirical risk minimization)

学习目标：在空间 $\mathcal{F}$ 中寻找能够在整个数据分布上表现最好的模型 f

$$\min_{f \in \mathcal{F}} \mathbb{E}_{(x,y) \sim D} [\text{loss}(f(x), y)] \quad \text{泛化风险}$$

现实任务中，无法得知完整的数据分布，只能获取训练数据

假设所有训练样本都是独立地从这个分布中采样而得

$$\min_{f \in \mathcal{F}} \sum_{i=1}^n \text{loss}(f(x_i), y_i) \quad \text{经验风险}$$

泛化风险 vs. 经验风险

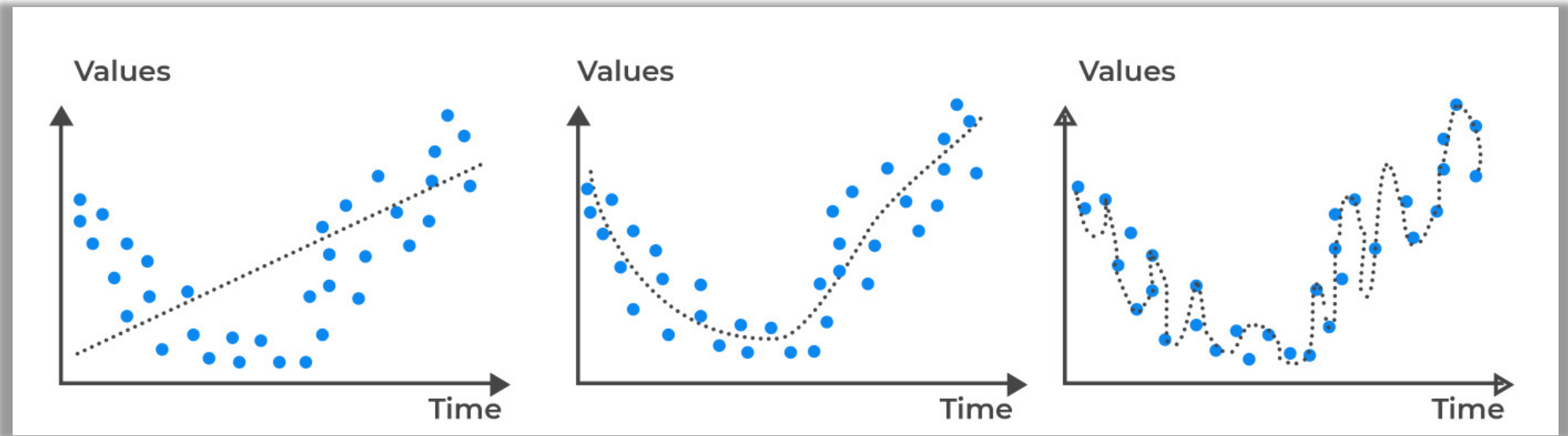
泛化误差：在“未来”样本上的误差

经验误差：在训练集上的误差，亦称“训练误差”

- 泛化误差越小越好
- 经验误差是否越小越好？

NO! 因为会出现“**过拟合**” (overfitting)

过拟合(overfitting) vs. 欠拟合(underfitting)



underfitting

Good fit

overfitting

过拟合(overfitting) vs 欠拟合(underfitting)



一般而言，训练样本越少，模型越复杂，越容易过拟合

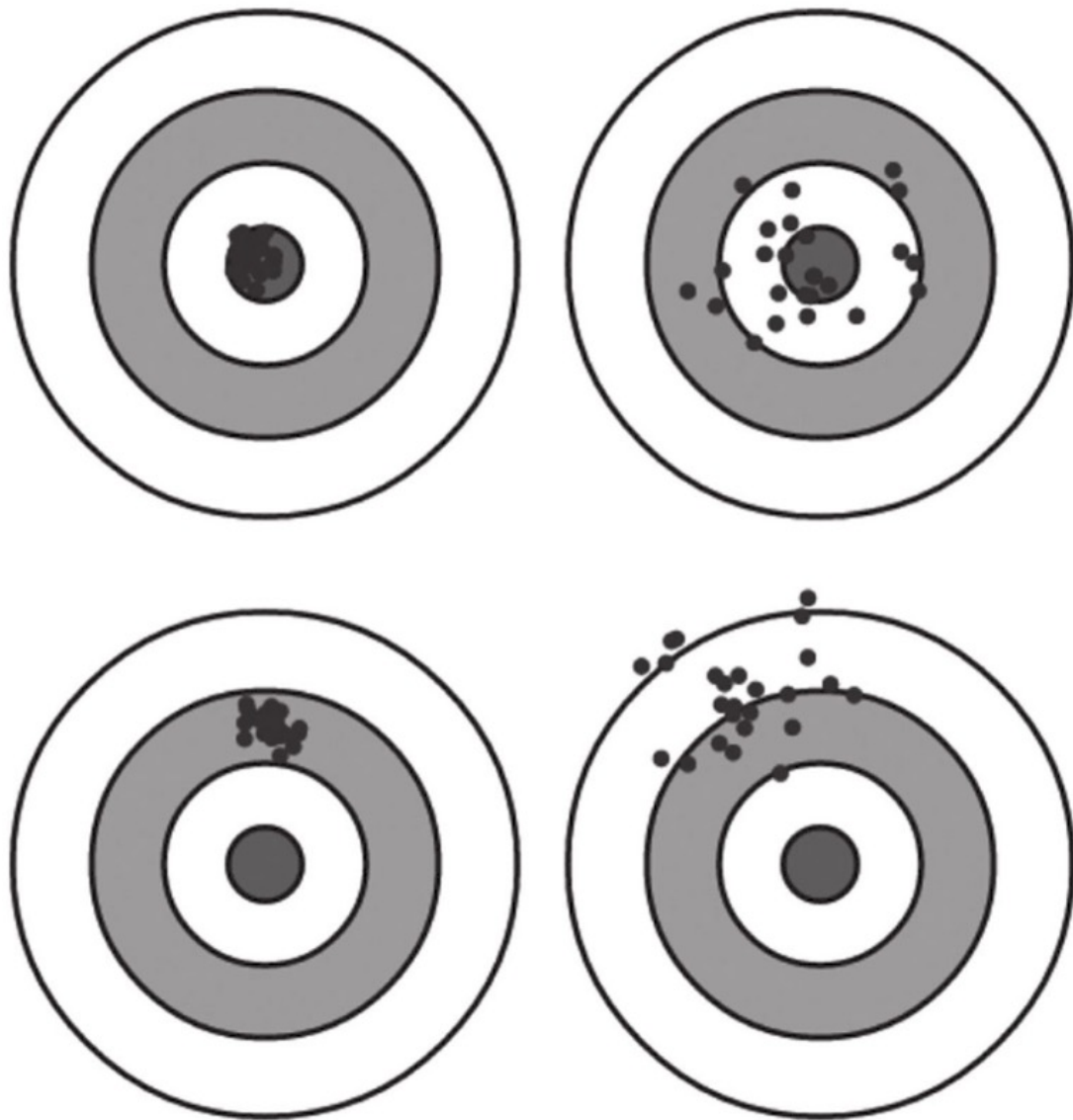
机器学习的误差

“误差” 包含了哪些因素？

换言之，从机器学习的角度看，

“误差” 从何而来？

机器学习的误差



偏差-方差分解 (bias-variance decomposition)

对回归任务，泛化误差可通过“偏差-方差分解”拆解为：

$$E(f; D) = \underbrace{bias^2(\mathbf{x})}_{\text{red}} + \underbrace{var(\mathbf{x})}_{\text{blue}} + \underbrace{\varepsilon^2}_{\text{green}}$$

期望输出与真实
输出的差别

$$bias^2(\mathbf{x}) = (\bar{f}(\mathbf{x}) - y)^2$$

同样大小的训练集的变
动，所导致的性能变化

$$var(\mathbf{x}) = \mathbb{E}_D \left[(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2 \right]$$

训练样本的标记与
真实标记有区别

表达了当前任务上任何学习算法
所能达到的期望泛化误差下界

$$\varepsilon^2 = \mathbb{E}_D \left[(y_D - y)^2 \right]$$

泛化性能是由学习算法的能力、数据的充分性以及学习任务本身的难度共同决定

偏差-方差困境 (bias-variance dilemma)

一般而言，偏差与方差存在冲突：

- 训练不足时，学习器拟合能力不强，偏差主导
- 随着训练程度加深，学习器拟合能力逐渐增强，方差逐渐主导
- 训练充足后，学习器的拟合能力很强，方差主导

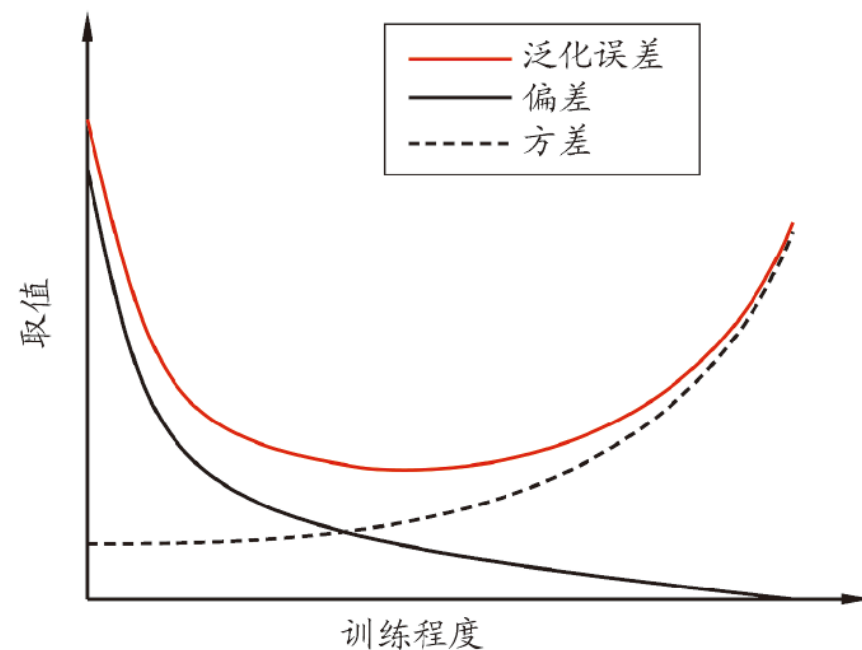
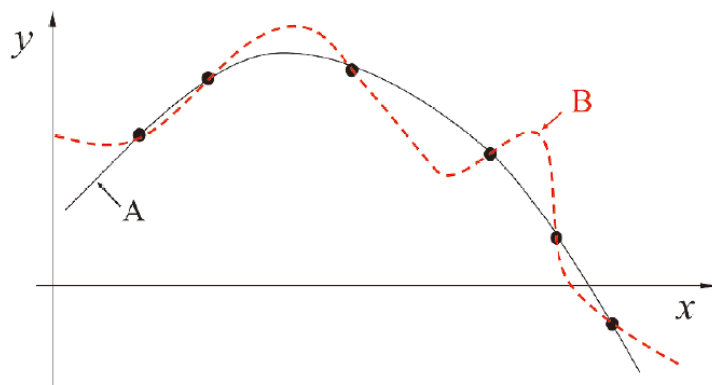


图 2.9 泛化误差与偏差、方差的关系示意图

归纳偏好(inductive bias)

机器学习算法在学习过程中对某种类型假设的偏好



A更好?
B更好?

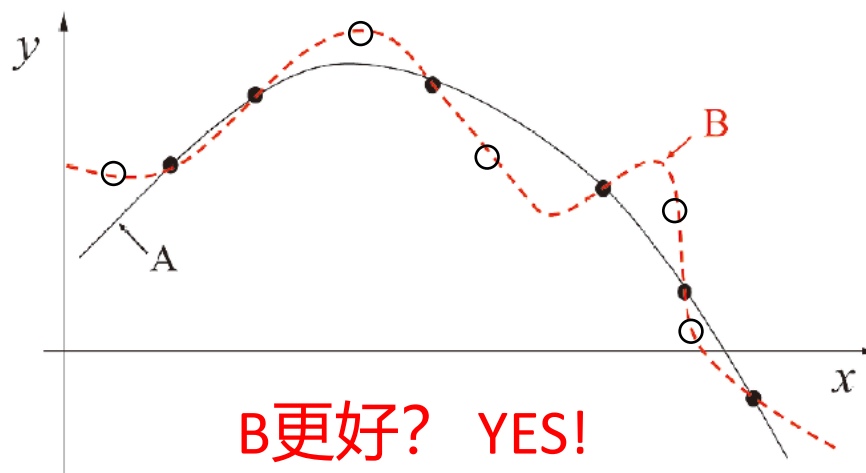
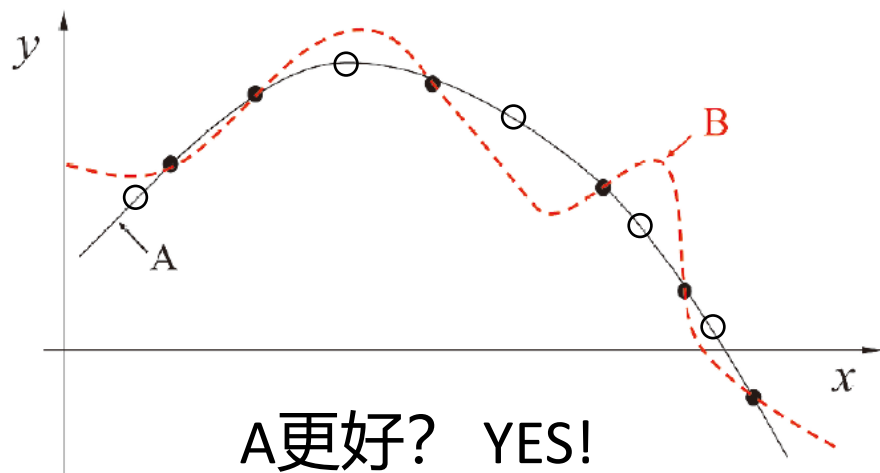
一般原则:
奥卡姆剃刀
(Occam's razor)

任何一个有效的机器学习算法必有其偏好

**学习算法的归纳偏好是否与问题本身匹配，
大多数时候直接决定了算法能否取得好的性能！**

哪个算法更好？

黑点：训练样本；白点：测试样本



没有免费的午餐!

No Free Lunch 定理：一个算法 $\mathcal{A}_1$ 若在某些问题上比另一个算法 $\mathcal{A}_2$ 好，必存在另一些问题， $\mathcal{A}_2$ 比 $\mathcal{A}_1$ 好。

NFL定理的寓意

NFL定理的重要前提：

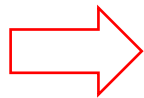
所有“问题”出现的机会相同、或所有问题同等重要

实际情形并非如此；我们通常只关注自己正在试图解决的问题

脱离具体问题，空泛地谈论“什么学习算法更好”
毫无意义！

具体问题，具体分析！

模型选择

□ 如何获得测试结果?  评估方法

评估方法

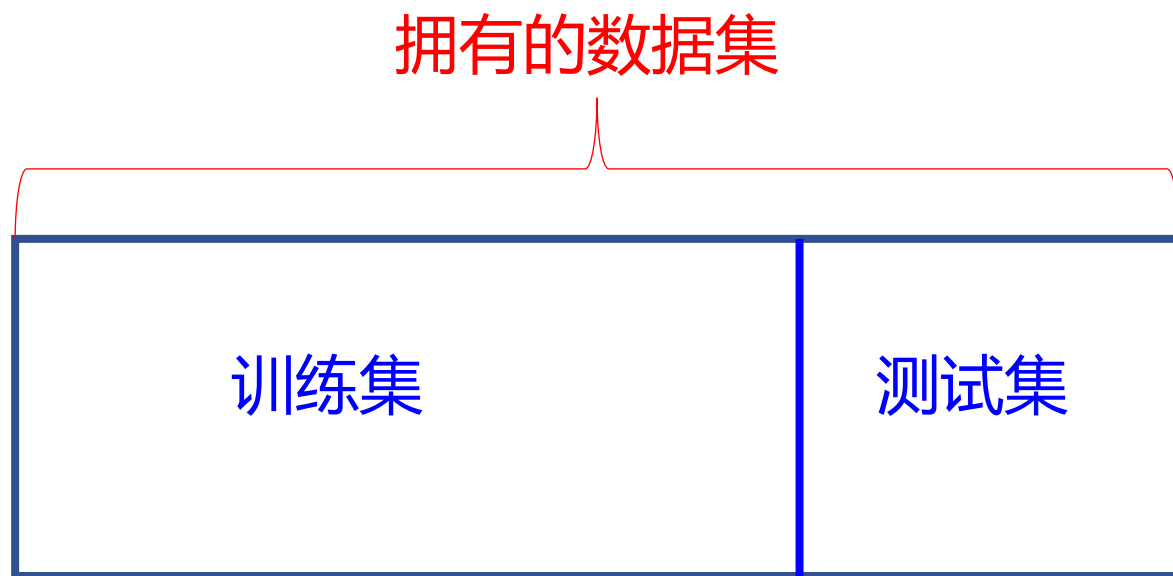
关键：怎么获得“测试集” (test set) ？

测试集应该与训练集 “互斥”

常见方法：

- 留出法 (hold-out)
- 交叉验证法 (cross validation)
- 自助法 (bootstrap)

留出法(Hold-out)



注意：

- 保持数据分布一致性（例如：分层采样）
- 测试集不能太大、不能太小（例如：1/5~1/3）
- 多次重复划分，计算均值+方差（例如：10次随机划分）

K-折交叉验证法

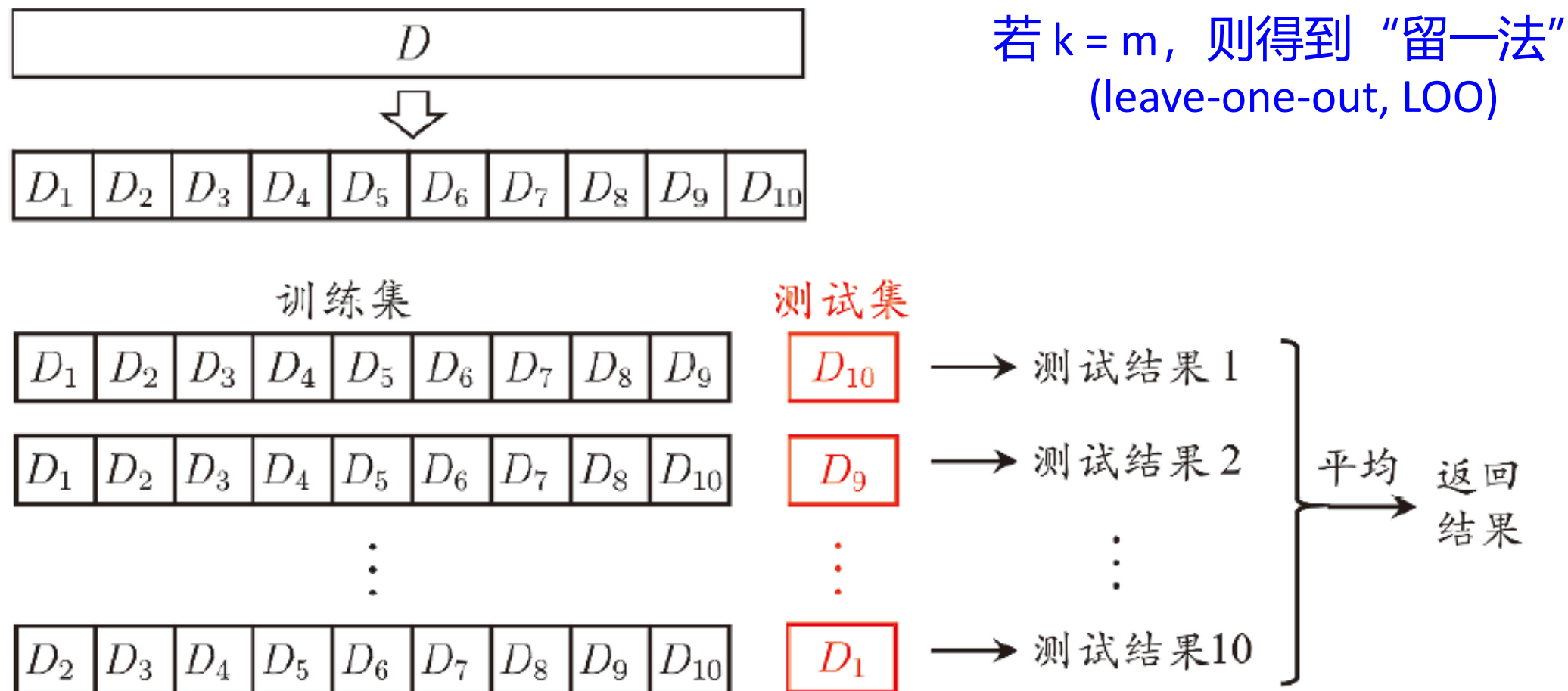
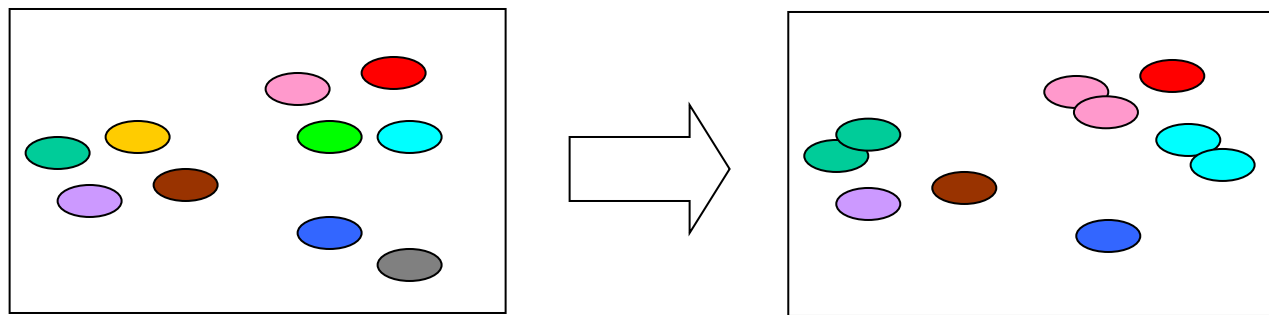


图 2.2 10 折交叉验证示意图

自助法

基于“自助采样” (bootstrap sampling)

亦称“有放回采样”、“可重复采样”



约有 36.8% 的样本不出现

$$\downarrow \lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m = \frac{1}{e} \approx 0.368$$

“包外估计” (out-of-bag estimation)

➤ 训练集与原样本集同规模

➤ 数据分布有所改变

“调参”与最终模型

算法的参数：一般由人工设定，亦称“超参数”

模型的参数：一般由学习确定

调参过程相似：先产生若干模型，然后基于某种评估方法进行选择

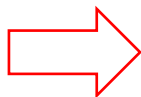
参数调得好不好对性能往往对最终性能有关键影响

区别：训练集 vs. 测试集 vs. 验证集 (validation set)

算法参数选定后，要用“训练集+验证集”重新训练最终模型

模型选择

□如何评估性能优劣？



性能度量

性能度量

性能度量(performance measure)是衡量模型泛化能力的评价标准，反映了任务需求

使用不同的性能度量往往会导致不同的评判结果

什么样的模型是“好”的，不仅取决于算法和数据，还取决于任务需求

□ 回归(regression) 任务常用均方误差：

$$E(f; D) = \frac{1}{m} \sum_{i=1}^m (f(\mathbf{x}_i) - y_i)^2$$

错误率与精度

□ 错误率:

$$E(f; D) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(\mathbf{x}_i) \neq y_i)$$

□ 精度:

$$\begin{aligned} \text{acc}(f; D) &= \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(\mathbf{x}_i) = y_i) \\ &= 1 - E(f; D) . \end{aligned}$$

查准率与查全率

表 2.1 分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

□ 查准率:
$$P = \frac{TP}{TP + FP}$$

□ 查全率:
$$R = \frac{TP}{TP + FN}$$

F1度量

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{\text{样例总数} + TP - TN}$$

比F1更一般的形式 F_β

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

$\beta = 1$: 标准F1

$\beta > 1$: 偏重查全率(逃犯信息检索)

$\beta < 1$: 偏重查准率(商品推荐系统)

机器学习是无所不能的吗？

并非“一切皆可学”，例如：

- ◆ 特征信息不充分

- 例如，重要特征信息没有获得

- ◆ 样本信息不充分

- 例如，仅有很少的数据样本

机器学习具有坚实的理论基础

计算学习理论

Computational learning theory

最重要的理论模型:

PAC (Probably Approximately Correct, 概率近似正确)

learning model

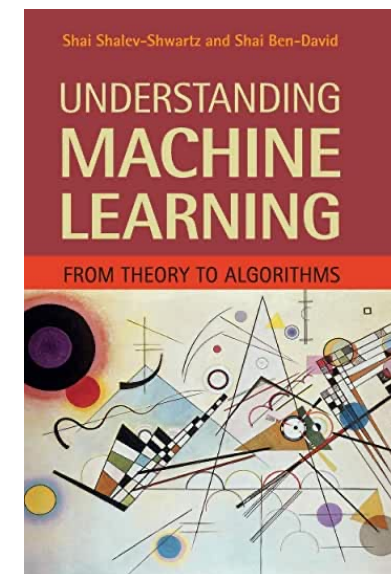
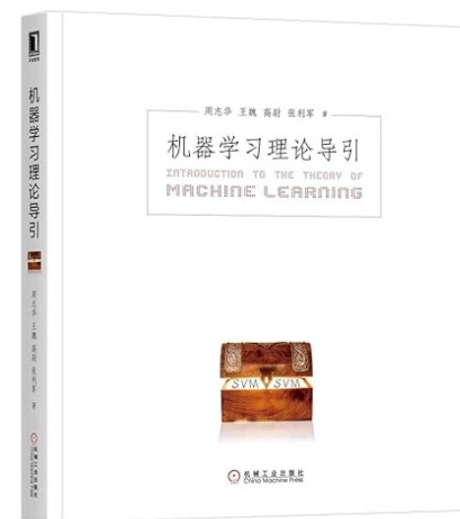
$$P(|f(\mathbf{x}) - y| \leq \epsilon) \geq 1 - \delta$$



Leslie Valiant
(莱斯利·维利昂特)
(1949-)
2010年图灵奖

机器学习具有坚实的理论基础

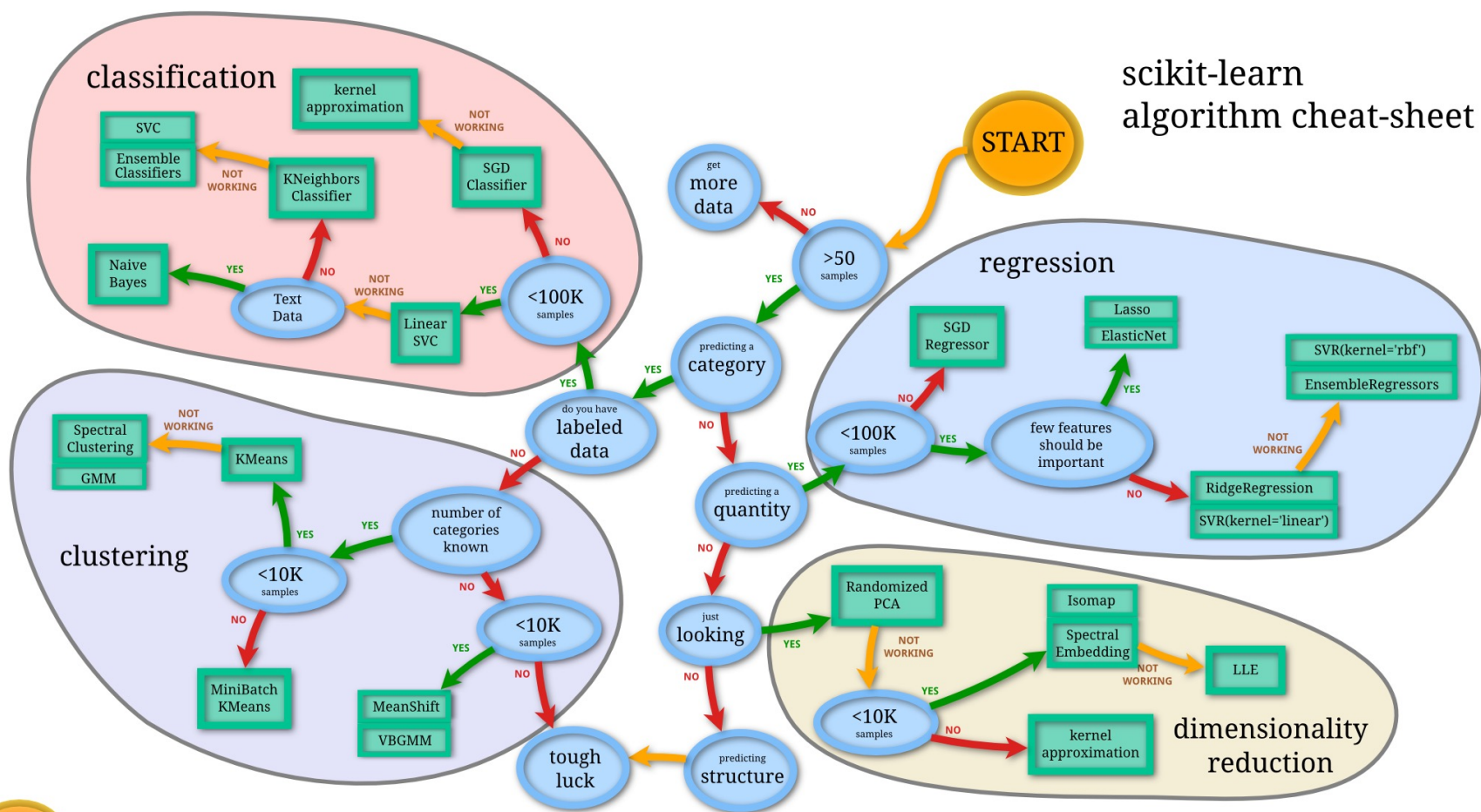
- 复杂度理论
- 泛化性理论
- 收敛性理论
- ...



机器学习常用算法

- 线性回归算法 (Linear Regression)
- 逻辑回归算法 (Logistic Regression)
- 支持向量机算法 (Support Vector Machine, SVM)
- k-近邻算法 (K-Nearest Neighbors, KNN)
- k-Means算法
- 决策树算法 (Decision Tree)
- 随机森林算法 (Random Forest)
- 朴素贝叶斯算法 (Naive Bayes)
- 神经网络 (Neural Network)
- ...

机器学习常用算法



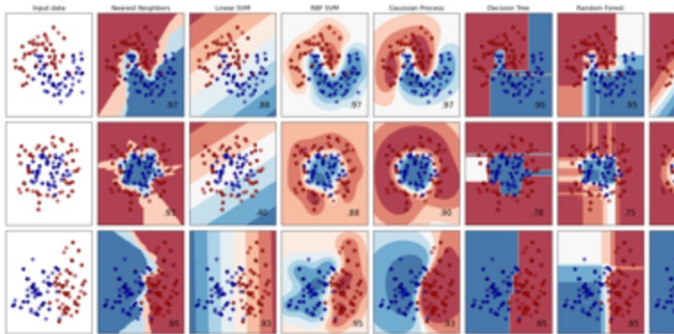
<https://scikit-learn.org/>

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

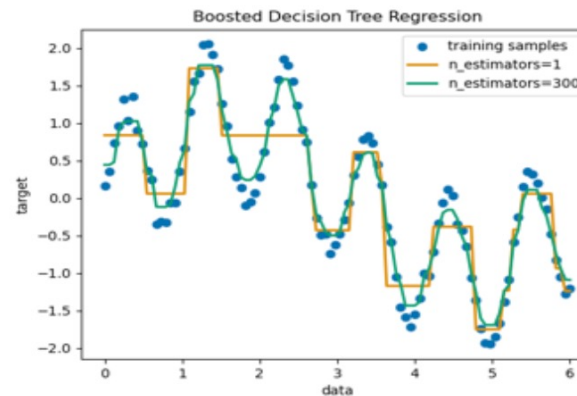


Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

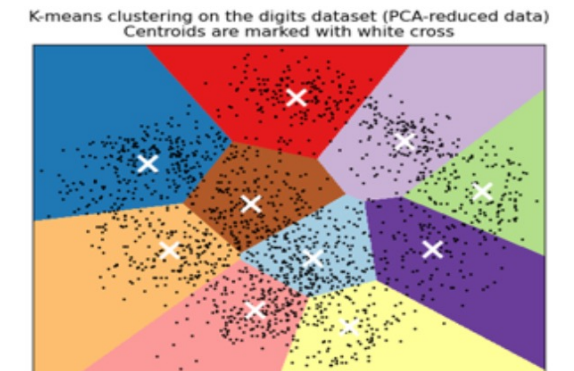


Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...



示例

训练阶段:

```
>>> import numpy as np
```

```
>>> from sklearn.linear_model import LinearRegression
```

```
>>> X = np.array([100, [110], [180]])
```

```
>>> y = np.array([300, [330], [540]])
```

```
>>> reg = LinearRegression().fit(X, y)
```

测试阶段:

```
>>> reg.predict(np.array([[140]]))
```

示例

```
In [12]: import numpy as np

In [13]: from sklearn.linear_model import LinearRegression

In [14]: X = np.array([[100], [110], [180]])

In [15]: y = np.array([[300], [330], [540]])

In [16]: reg = LinearRegression().fit(X,y)

In [17]: reg.predict(np.array([[140]]))
Out[17]: array([[420.]])

In [18]:
```

现实机器学习应用

把机器学习的“十大算法” “二十大算法” 都弄熟，
逐个试一遍，是否就“止于至善”了？

NO !

机器学习并非“十大套路” “二十大招数” 的简单堆积

现实任务千变万化，

以有限的“套路”应对无限的“问题”，焉有不败？

最优方案往往来自：**按需设计、度身定制**