

机器学习导论-模型评估与选择

Introduction to Machine Learning-Model Evaluation and Selection

李文斌

<https://cs.nju.edu.cn/liwenbin>

liwenbin@nju.edu.cn

2024年03月05日

统计学习方法三要素

□ 统计学习方法构成

方法 = 模型 + 策略 + 算法

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **模型 (model)** : 是所要学习的条件概率分布或决策函数或映射函数

□ **假设空间 (hypothesis space)** : 包含所有可能的条件概率分布或决策函数集合

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **模型 (model)** : 是所要学习的条件概率分布或决策函数或映射函数

□ **假设空间 (hypothesis space)** : 包含所有可能的条件概率分布或决策函数集合

$$\mathcal{F} = \{f | Y = f(X)\}$$

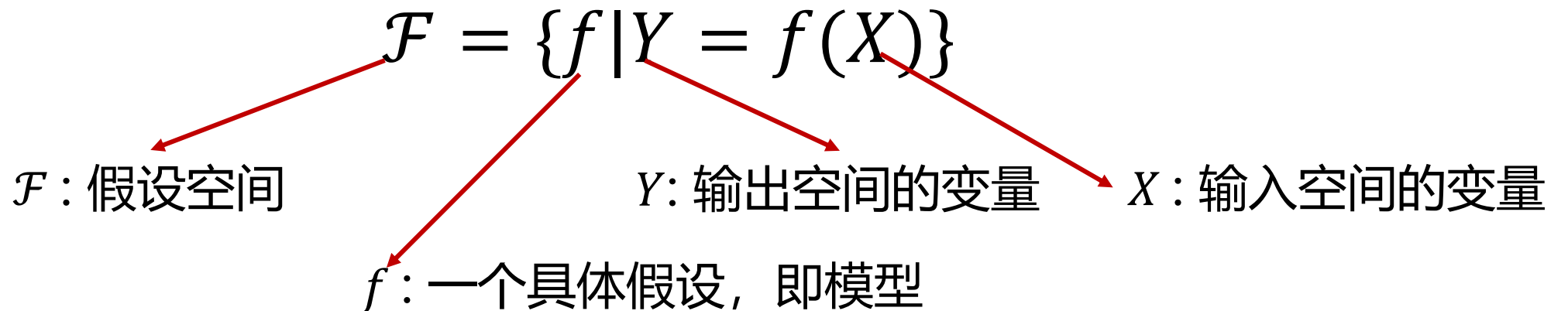
统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ 模型（model）：是所要学习的条件概率分布或决策函数或映射函数

□ 假设空间（hypothesis space）：包含所有可能的条件概率分布或决策函数集合



统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□模型（model）：是所要学习的条件概率分布或决策函数或映射函数

□假设空间（hypothesis space）：包含所有可能的条件概率分布或决策函数集合

$$\mathcal{F} = \{f | Y = f_{\theta}(X), \theta \in \mathbb{R}^n\}$$

f_{θ} ：假设/模型由参数向量 θ 刻画，取值于 n 维欧氏空间 \mathbb{R}^n ，也称为参数空间

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **策略 (strategy)**：如何从假设空间中选择最优的假设/模型

□ **损失函数 (loss function) 或者代价函数 (cost function)**

- 度量模型一次预测的好坏，记为 $L(Y, f(X))$
- 通常为一个非负实值函数

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **策略（strategy）**：如何从假设空间中选择最优的假设/模型

□ **损失函数（loss function）或者代价函数（cost function）**

□ **常见损失函数：**

❖ **0-1损失函数（0-1 loss function）**

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases}$$

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **策略（strategy）**：如何从假设空间中选择最优的假设/模型

□ **损失函数（loss function）或者代价函数（cost function）**

□ **常见损失函数：**

❖ **平方损失函数（quadratic loss function）**

$$L(Y, f(X)) = (Y - f(X))^2$$

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **策略 (strategy)**：如何从假设空间中选择最优的假设/模型

□ **风险函数 (risk function) 或期望损失 (expected loss)**

- 度量平均意义下模型预测的好坏，记为 $R_{exp}(f) = E[L(Y, f(X))]$
- 遗憾的是期望损失并不能直接计算

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ **策略** (strategy) : 如何从假设空间中选择最优的假设/模型

□ **经验风险** (empirical risk) : 模型在训练数据集上的平均损失, 记为 $R_{emp}(f)$

❖ 给定一个训练数据集

$$T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$$

❖ **经验风险**

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(\mathbf{x}_i))$$

根据大数定律, 当样本数量 N 趋向于无穷时, 经验风险趋于期望风险

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ 经验风险最小化（empirical risk minimization, ERM）：

- 经验风险最小的模型就是最优的模型
- 即求解最小化的优化问题：

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N L(y_i, f(\mathbf{x}_i))$$

有时候会出现过拟合（over-fitting）现象，例如样本量很少的情况下

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ 结构风险最小化（structural risk minimization, SRM）：

- 结构风险最小的模型就是最优的模型
- 增加衡量模型复杂度的正则化项（regularizer）或罚项（penalty term）
- 即求解最小化的优化问题：

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N L(y_i, f(\mathbf{x}_i)) + \lambda \cdot J(f)$$

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

方法 = 模型 + 策略 + 算法

□ 结构风险最小化（structural risk minimization, SRM）：

- 结构风险最小的模型就是最优的模型
- 增加衡量模型复杂度的正则化项（regularizer）或罚项（penalty term）
- 即求解最小化的优化问题：

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N L(y_i, f(\mathbf{x}_i)) + \lambda \cdot J(f)$$

λ : 正则化系数, 超参数

$J(f)$: 模型复杂度

统计学习方法三要素

□ 统计学习方法构成（监督学习为例）

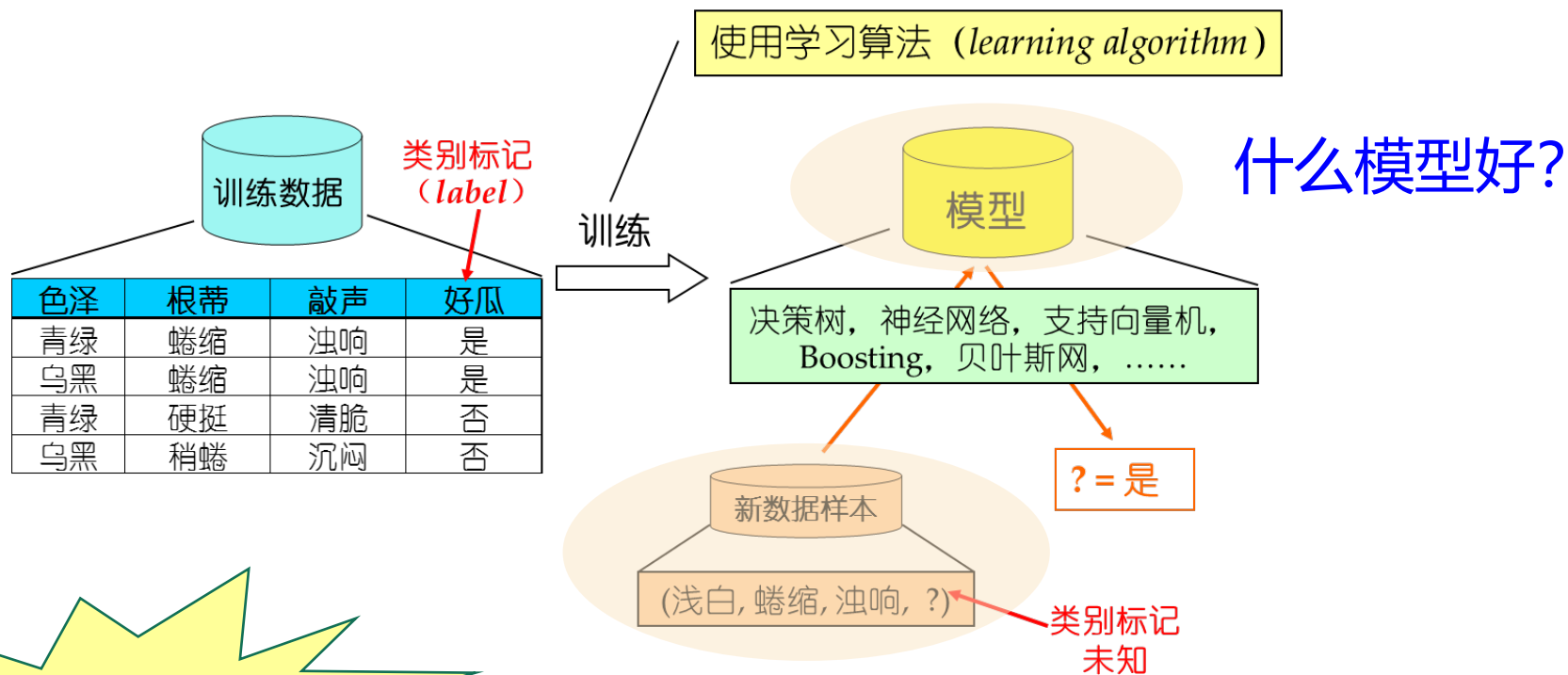
方法 = 模型 + 策略 + 算法

□ **算法 (algorithm)**：学习模型的具体计算方法

- 统计学习问题往往形式化为一个**最优化问题**
- 如果最优化问题**有解析解/闭式解 (closed-form expression)**，直接求解
- 如果最优化问题**没有有解析解/闭式解 (closed-form expression)**
 - 梯度下降 (gradient descent)
 - 牛顿法 (Newton method)
 - 拟牛顿法 (quasi-Newton method)
 - ...

模型评估与选择

□ 典型的机器学习过程



泛化能力强!

能很好地适用于 unseen instance

例如, 错误率低、精度高

然而, 我们手上没有 unseen instance,

模型评估与选择

□ 泛化误差 vs. 经验误差

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)

模型评估与选择

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

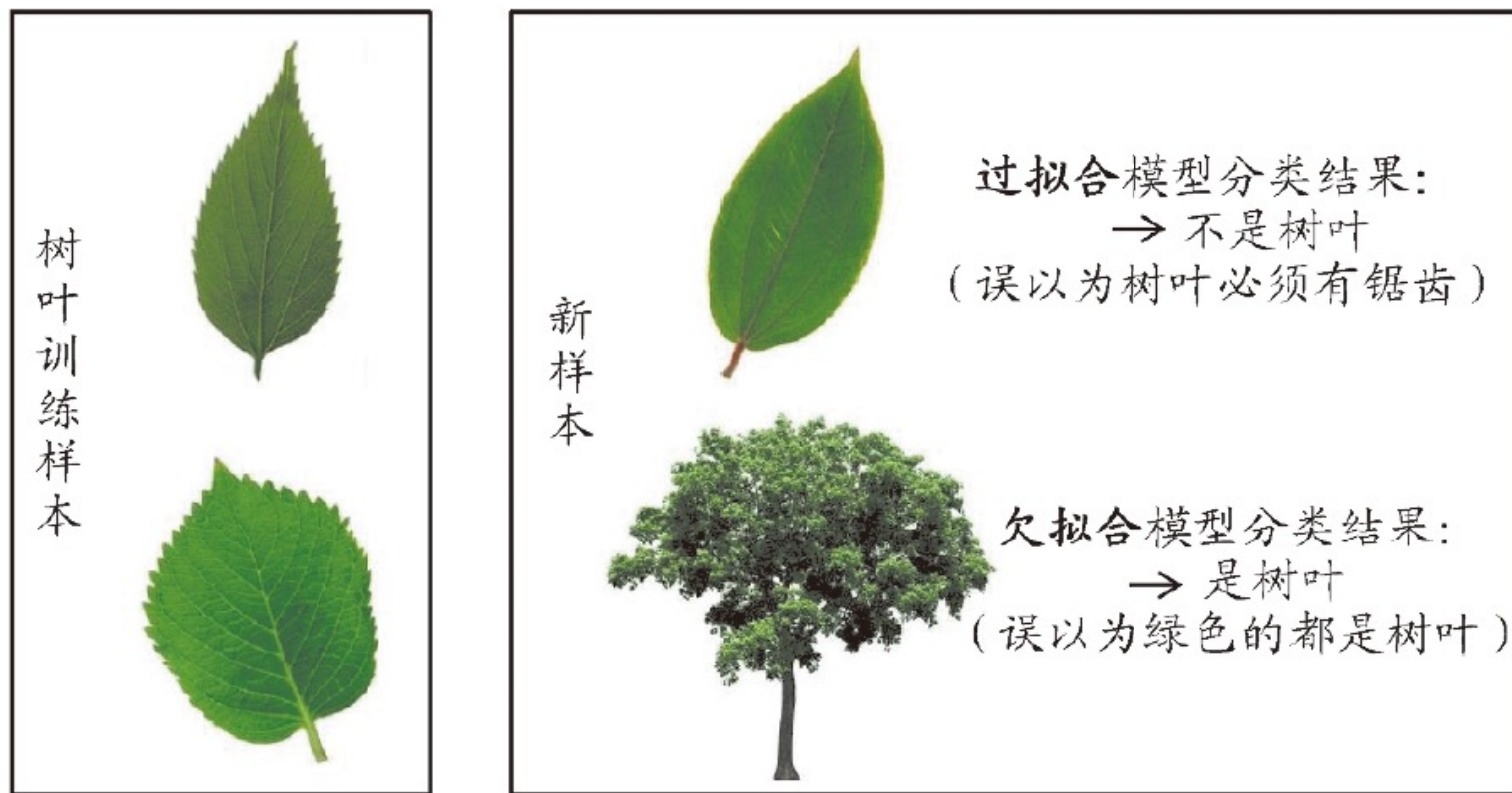


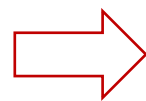
图 2.1 过拟合、欠拟合的直观类比

模型评估与选择

□ 模型选择 (model selection)

三个关键问题:

□ 如何获得测试结果?



评估方法

□ 如何评估性能优劣?



性能度量

□ 如何判断实质差别?



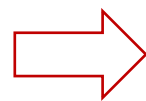
比较检验

模型评估与选择

□ 模型选择 (model selection)

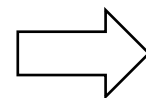
三个关键问题:

□ 如何获得测试结果?



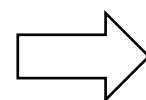
评估方法

□ 如何评估性能优劣?



性能度量

□ 如何判断实质差别?



比较检验

模型评估与选择

□ 评估方法

关键：怎么获得“测试集”(test set)？

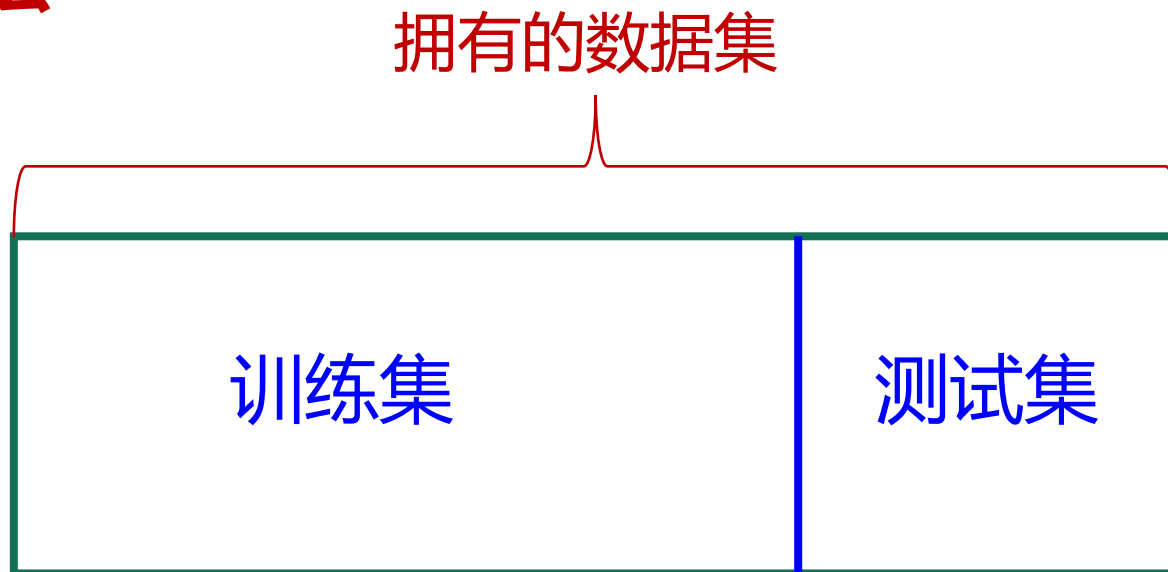
测试集应该与训练集 “互斥”

常见方法：

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

评估方法

□ 留出法



注意：

- 保持数据分布一致性（例如：分层采样）
- 多次重复划分（例如：100次随机划分）
- 测试集不能太大、不能太小（例如：1/5~1/3）

评估方法

□ K-折交叉验证法

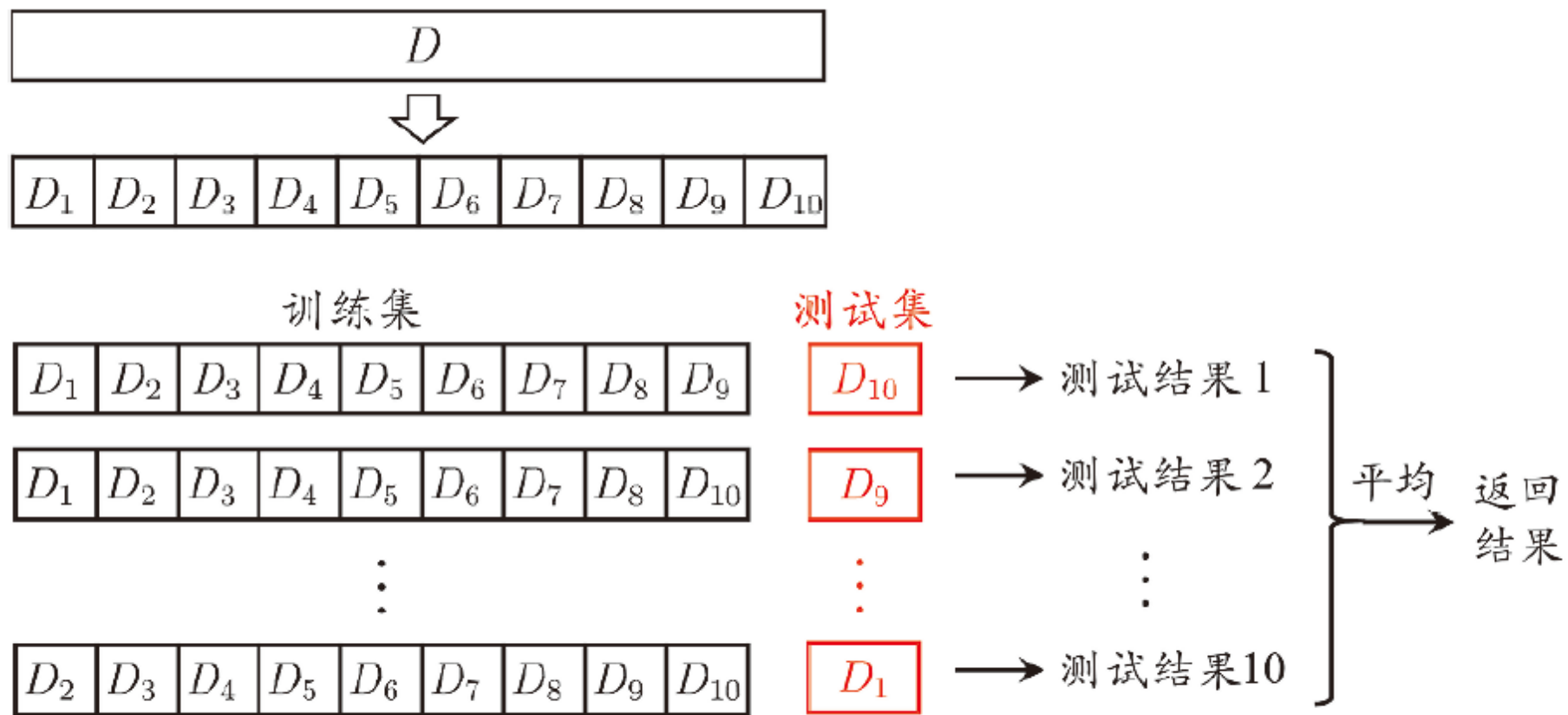


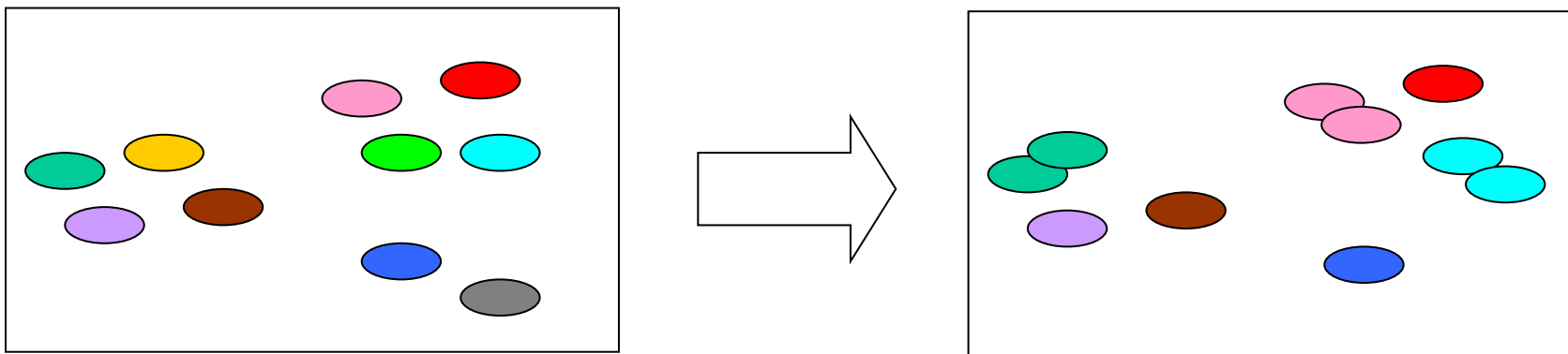
图 2.2 10 折交叉验证示意图

评估方法

□ 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

模型评估与选择

□ “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

模型评估与选择

□ 模型选择 (model selection)

三个关键问题:

□ 如何获得测试结果? ⇒ 评估方法

□ 如何评估性能优劣? ⇒ 性能度量

□ 如何判断实质差别? ⇒ 比较检验

模型评估与选择

□ 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：

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

性能度量

□ 错误率 vs. 精度

□ 错误率:

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

□ 精度:

$$acc(f; D) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(x_i) = y_i) = 1 - E(f; D)$$

性能度量

□ 查准率 vs. 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: $R = \frac{TP}{TP + FN}$

性能度量

□ F1度量

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

$$\frac{1}{F1} = \frac{1}{2} \cdot \left(\frac{1}{P} + \frac{1}{R} \right)$$

若对查准率/查全率有不同偏好:

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

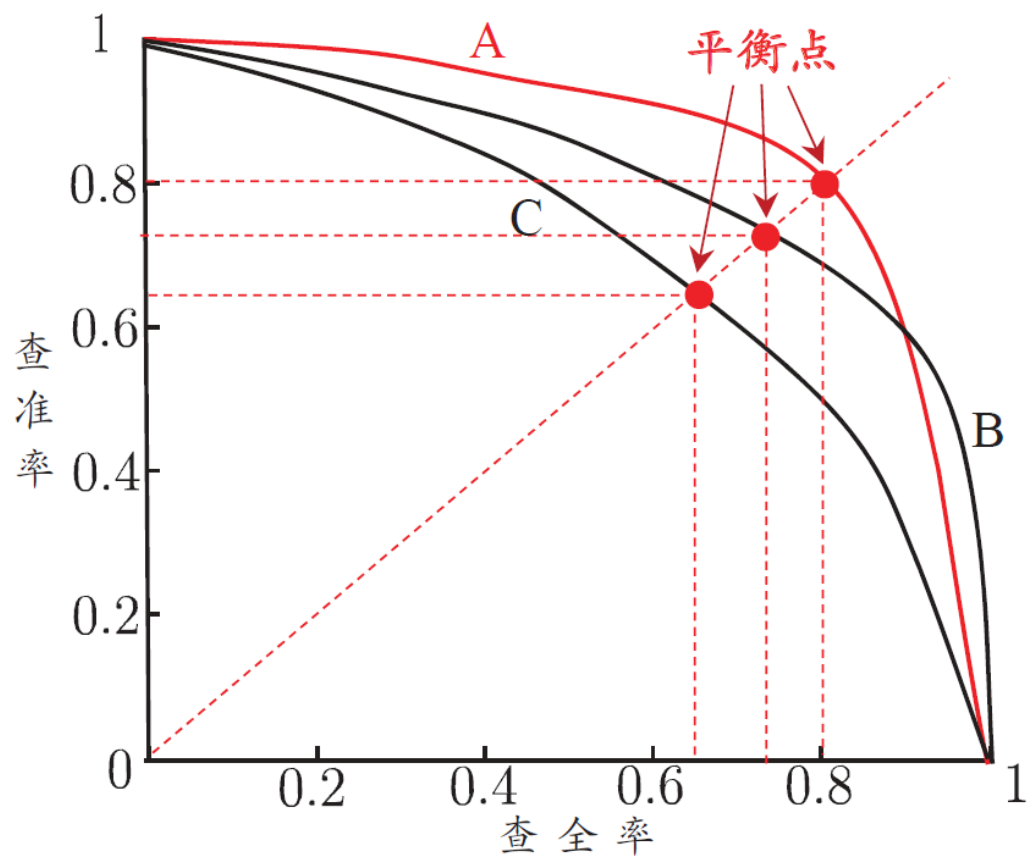
$$\frac{1}{F_{\beta}} = \frac{1}{1 + \beta^2} \cdot \left(\frac{1}{P} + \frac{\beta^2}{R} \right)$$

$\beta > 1$ 时查全率有更大影响; $\beta < 1$ 时查准率有更大影响

性能度量

□ P-R曲线图, BEP(Break-Event Point, 平衡点)

根据学习器的预测结果按正例可能性大小对样例进行排序, 并逐个把样本作为正例进行预测



P-R图:

- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C
- 学习器 A ?? 学习器 B

BEP:

- 学习器 A 优于 学习器 B
- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C

性能度量

□ 宏XX vs. 微XX

若能得到多个混淆矩阵:

(例如多次训练/测试的结果, 多分类的两两混淆矩阵)

宏(macro-)查准率、查全率、F1

$$macro - P = \frac{1}{n} \sum_{i=1}^n P_i$$

$$macro - R = \frac{1}{n} \sum_{i=1}^n R_i$$

$$macro - F1 = \frac{2 \times macro - P \times macro - R}{macro - P + macro - R}$$

微(micro-)查准率、查全率、F1

$$micro - P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}$$

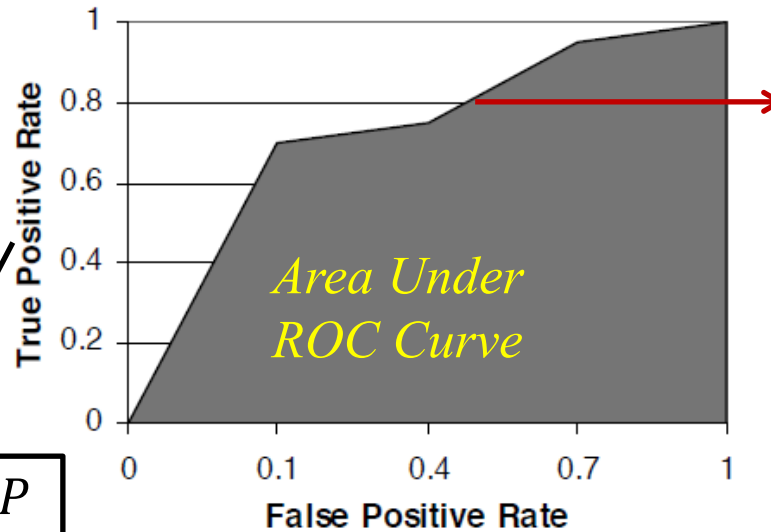
$$micro - R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}$$

$$micro - F1 = \frac{2 \times micro - P \times micro - R}{micro - P + micro - R}$$

性能度量

□ ROC, AUC

AUC: Area Under the ROC Curve



ROC (Receiver Operating Characteristic) Curve

[Green & Swets, Book 66; Spackman, IWML'89]

The bigger, the better

$$tpr = \frac{TP}{TP + FN} = \frac{TP}{m_+}$$

$$fpr = \frac{FP}{FP + TN} = \frac{FP}{m_-}$$

$$AUC = 1 - \frac{1}{m^+ m^-} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} \left(\mathbb{I}(f(x^+) < f(x^-)) + \frac{1}{2} \mathbb{I}(f(x^+) = f(x^-)) \right)$$

性能度量

□ 非均等代价

犯不同的错误往往会造成不同的损失

此时需考虑“非均等代价” (unequal cost)

表 2.2 二分类代价矩阵

真实类别	预测类别	
	第 0 类	第 1 类
第 0 类	0	$cost_{01}$
第 1 类	$cost_{10}$	0

□ 代价敏感(cost-sensitive)错误率:

$$E(f; D; \text{cost}) = \frac{1}{m} \left(\sum_{x_i \in D^+} \mathbb{I}(f(x_i) \neq y_i) \times \text{cost}_{01} + \sum_{x_i \in D^-} \mathbb{I}(f(x_i) \neq y_i) \times \text{cost}_{10} \right)$$

模型评估与选择

□ 模型选择 (model selection)

三个关键问题:

□ 如何获得测试结果? ⇒ 评估方法

□ 如何评估性能优劣? ⇒ 性能度量

□ 如何判断实质差别? ⇒ 比较检验

模型评估与选择

□ 比较检验

在某种度量下取得评估结果后，是否可以直接比较以评判优劣？

- NO ! 因为：**
- 测试性能不等于泛化性能
 - 测试性能随着测试集的变化而变化
 - 很多机器学习算法本身有一定的随机性

机器学习 “概率近似正确”

比较检验

□ 常用方法

统计假设检验 (hypothesis test) 为学习器性能比较提供了重要依据

□ 两学习器比较

- 交叉验证 t 检验 (基于成对 t 检验)
 - k 折交叉验证; 5x2交叉验证
- McNemar 检验 (基于列联表, 卡方检验)



统计显著性

□ 多学习器比较

- Friedman + Nemenyi
 - Friedman检验 (基于序值, F检验; 判断“是否都相同”)
 - Nemenyi 后续检验 (基于序值, 进一步判断两两差别)

比较检验

□ Friedman检验图

横轴为平均序值，每个算法圆点为其平均序值，线段为临界阈值的大小

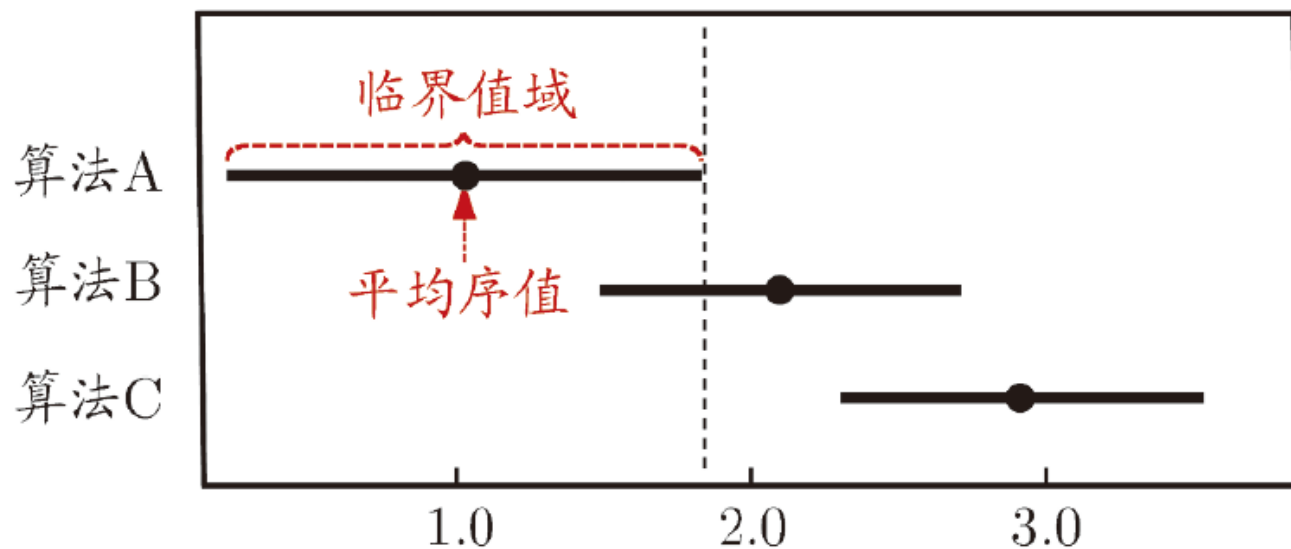


图 2.8 Friedman 检验图

若两个算法有交叠 (A 和 B)，则说明没有显著差别；
否则有显著差别 (A 和 C)，
算法 A 显著优于算法 C

模型评估与选择

“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？

模型评估与选择

□ 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：

$$E(f; D) = \underbrace{bias^2(x)}_{\text{期望输出与真实输出的差别}} + \underbrace{var(x)}_{\text{同样大小的训练集的变动, 所导致的性能变化}} + \underbrace{\varepsilon^2}_{\text{训练样本的标记与真实标记有区别}}$$

期望输出与真实
输出的差别

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

同样大小的训练集的
变动, 所导致的
性能变化

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

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

$$\varepsilon^2 = E_D[(y_D - y)^2]$$

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

模型评估与选择

□ 偏差-方差窘境 (bias-variance dilemma/tradeoff)

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

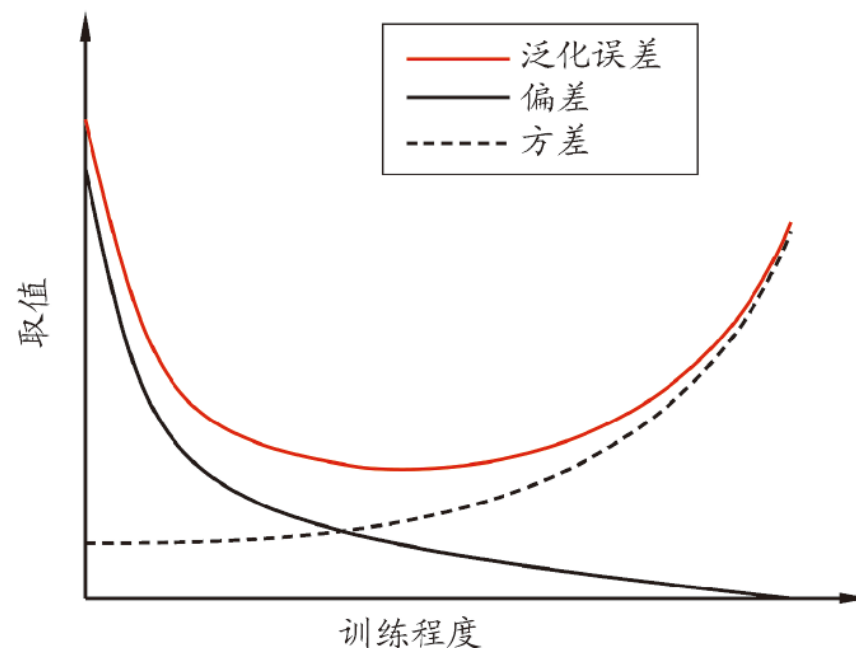


图 2.9 泛化误差与偏差、方差的关系示意图

模型评估与选择

□ 偏差-方差窘境 (bias-variance dilemma/tradeoff)

□ Bias (学习结果的期望与真实规律的差距)

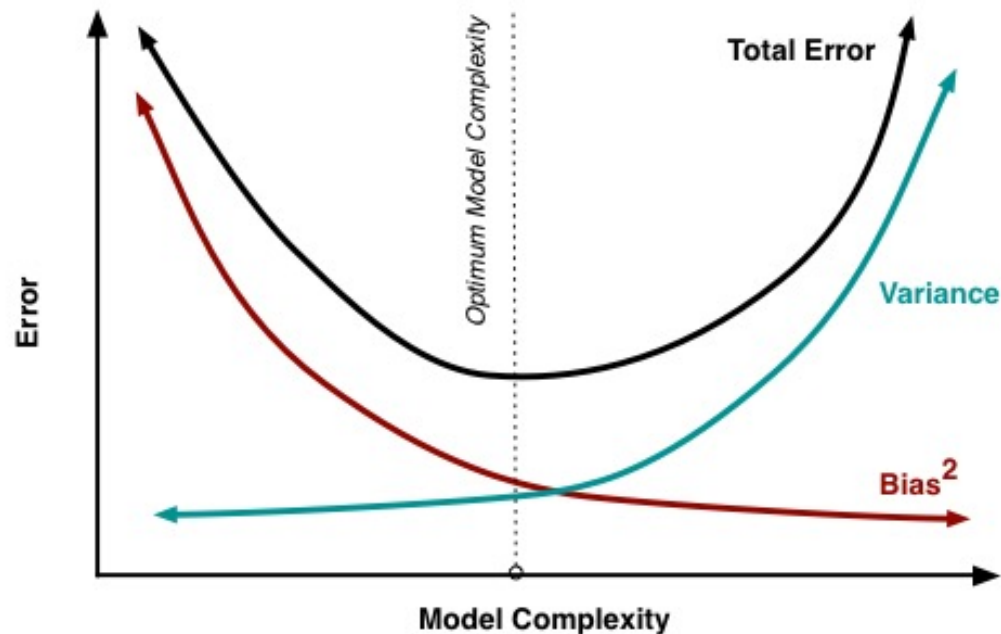
$$Bias = E[\hat{f}(x)] - f(x)$$

□ Variance (学习结果自身的不稳定性)

$$Variance = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

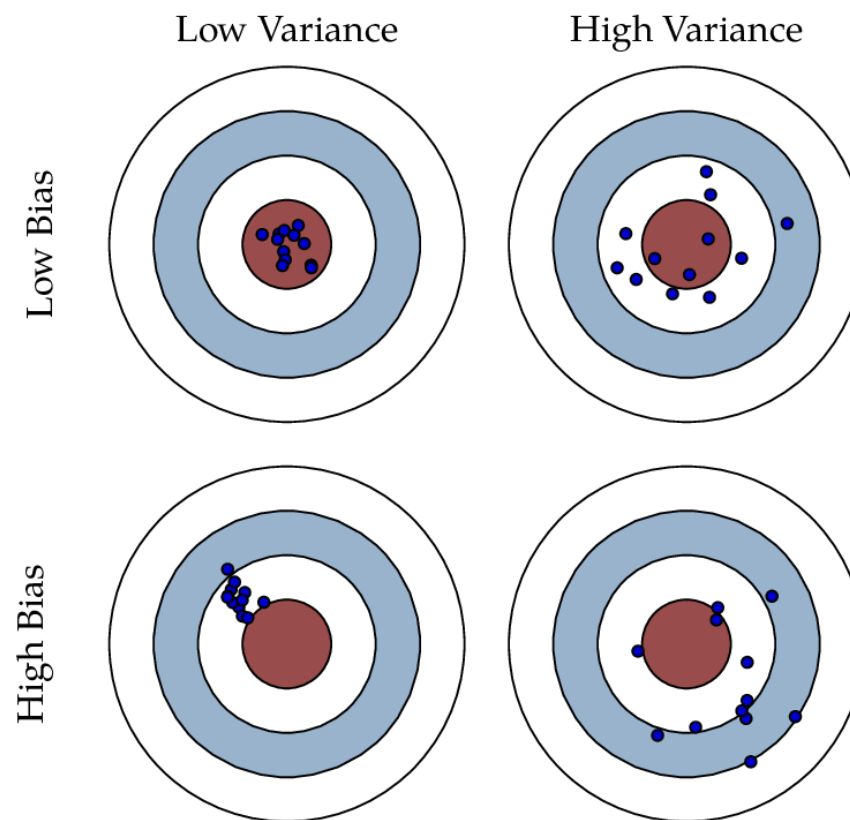
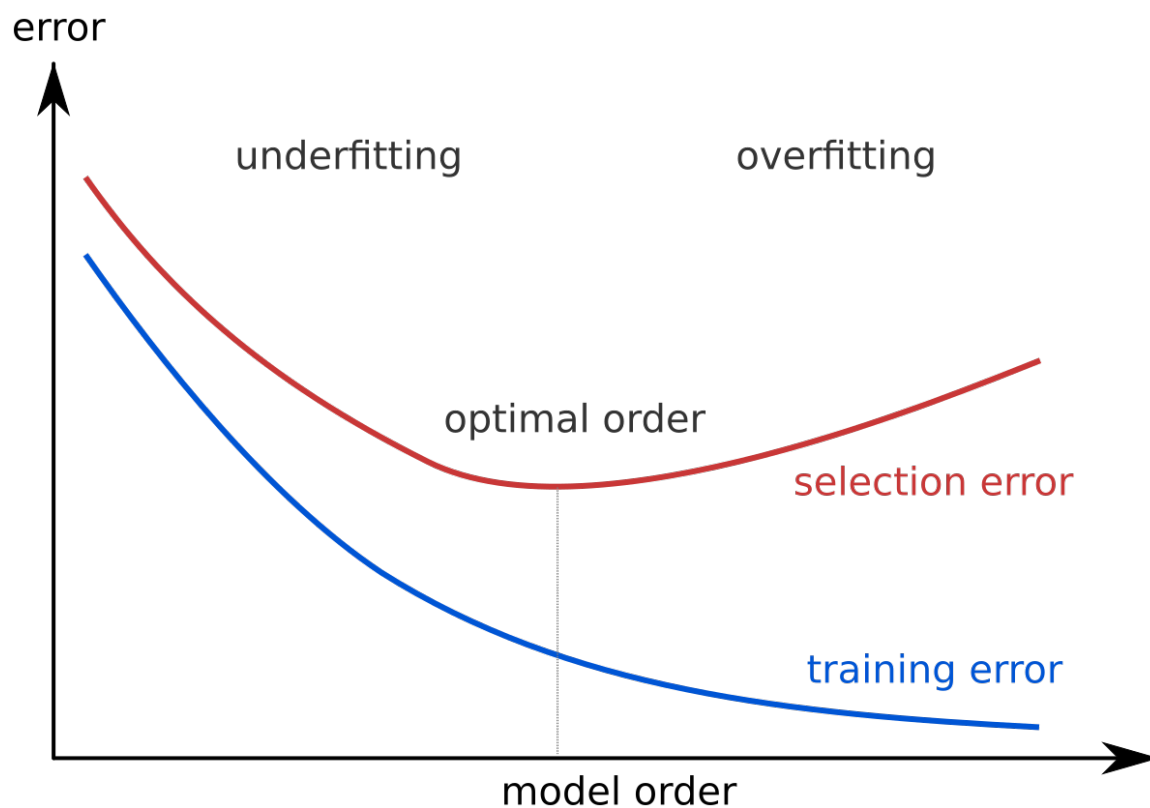
□ Total Error (以均方误差为例)

$$Err(x) = Bias^2 + Variance + Random Error$$



模型评估与选择

□ 偏差-方差窘境 (bias-variance dilemma/tradeoff)



模型评估与选择

□ 如何减少高偏差 (High bias)

- 结合数据其他的特征，来提高模型的精度
- 增加训练的迭代次数，使得模型可以学习更加复杂的数据
- 避免使用高偏差的算法，如线性回归、逻辑回归、判别分析等，而是使用非线性算法，如 K -近邻、支持向量机、决策树等
- 在不同程度上减少正则化，以帮助模型更有效地学习训练集，并防止欠拟合

模型评估与选择

□ 如何减少高方差 (High variance)

- 减少模型中特征的数量
- 将当前模型替换为更简单的模型
- 增加训练数据的多样性以平衡模型的复杂度和数据结构
- 避免使用高方差算法（支持向量机、决策树、K近邻等），而选择低方差算法，如线性回归、逻辑回归和线性判别分析
- 进行超参数调优以避免过拟合
- 增加输入的正则化以减少模型的复杂度并防止过拟合
- 使用新的模型架构（如果其他方法不起作用，应将其视为最后的手段）

谢谢!

联系方式: liwenbin@nju.edu.cn

更多信息: <https://cs.nju.edu.cn/liwenbin>

Sk-Learn的安装和使用

Installation and Usage of **Scikit-learn**

助教：陈恽飏

522023330016@smail.nju.edu.cn

2024年3月5日

目录

- Scikit-learn简介
- 环境安装
- 使用示例
- 相关资源

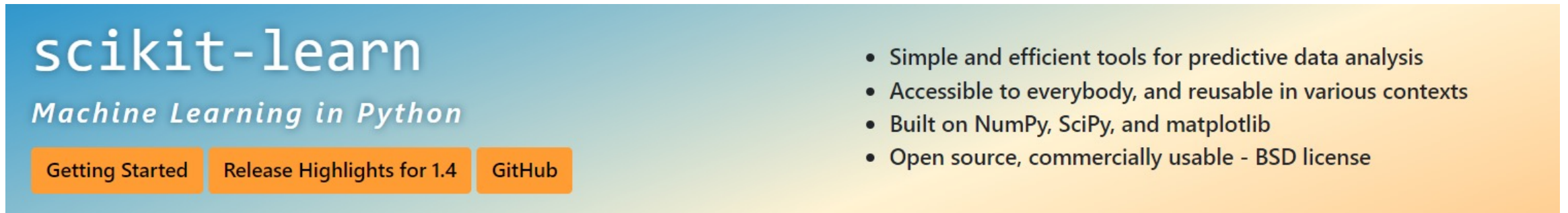
❑ Scikit-learn简介

❑ 环境安装

❑ 使用示例

❑ 相关资源

Scikit-learn简介



- ❑ Scikit-learn (简称sklearn) 是一个**开源的Python语言的机器学习库**
 - ❑ 基于NumPy, SciPy和matplotlib构建
- ❑ 为机器学习提供强大支持，提供了一系列**高效的机器学习和统计建模工具**
 - ❑ 包括但不限于**分类、回归、聚类和降维**功能。
- ❑ 接口简洁易使用，使得机器学习算法的评估和调参变得简单直观，API的文档支持好
- ❑ 拥有 BSD 许可证，因此开发者可以将其无限制地用于商业应用

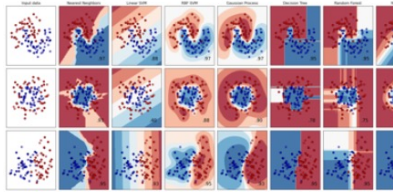
Scikit-learn简介

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

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



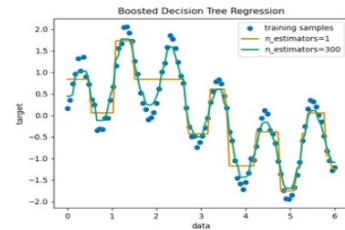
Examples

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...



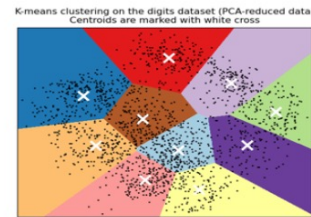
Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

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



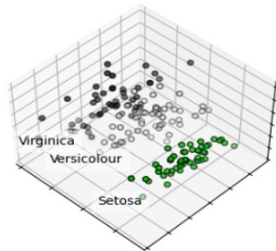
Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization, and more...



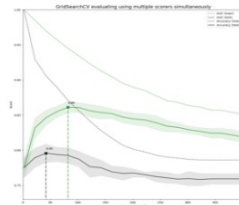
Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross validation, metrics, and more...



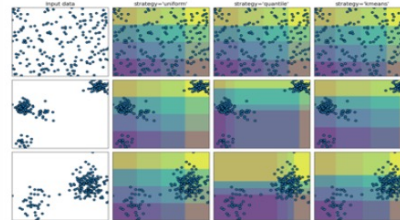
Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...



Examples

- 分类
- 回归
- 聚类
- 降维
- 模型选择
- 预处理
- ...

□ Scikit-learn简介

□ **环境安装**

□ 使用示例

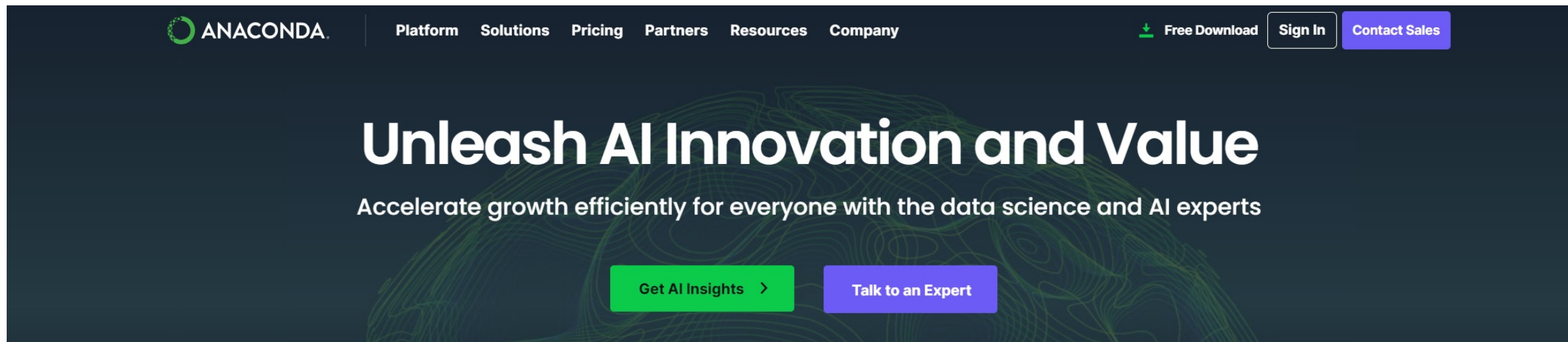
□ 相关资源

环境安装



□ 推荐使用Anaconda作为环境包管理工具

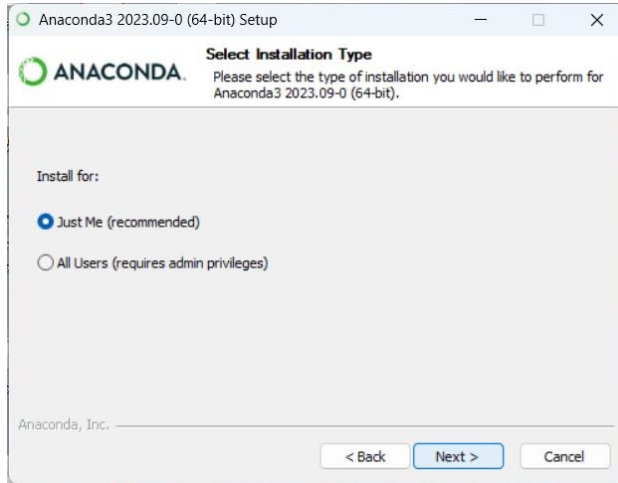
- 预装了超过1,500个数据科学相关的包，包括常用的NumPy、Pandas、SciPy、Matplotlib、Jupyter Notebook
- 同时允许创建、导出、列出、删除和更新不同的环境，这些环境可以包含不同版本的包和Python版本，从而避免不同版本之间的冲突。



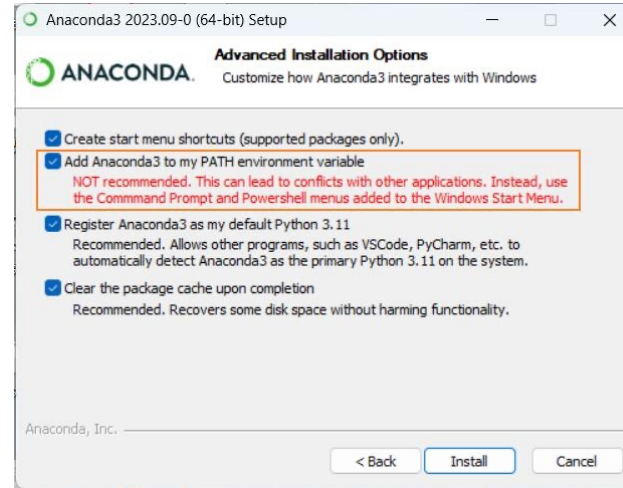
环境安装



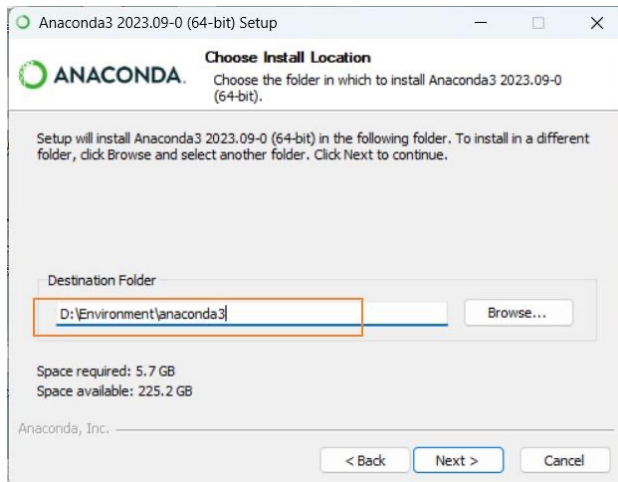
1.



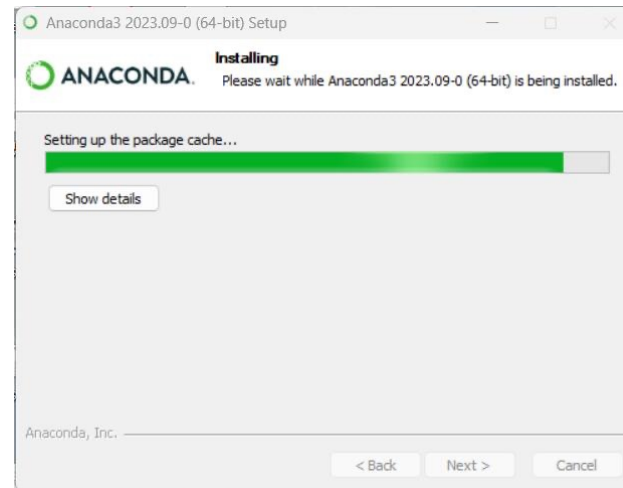
3.



2.



4.



5.



环境安装

在安装完anaconda之后，进入命令行cmd：输入如下指令

❑ 创建conda环境

```
conda create -n ml_lib python=3.10
```



❑ 切换到新的环境下

```
conda activate ml_lib
```

❑ 安装sk-learn

```
conda install scikit-learn
```

❑ 安装matplotlib (画图)

```
conda install matplotlib
```

❑ 安装notebook (交互式环境)

```
conda install notebook
```

□ Scikit-learn简介

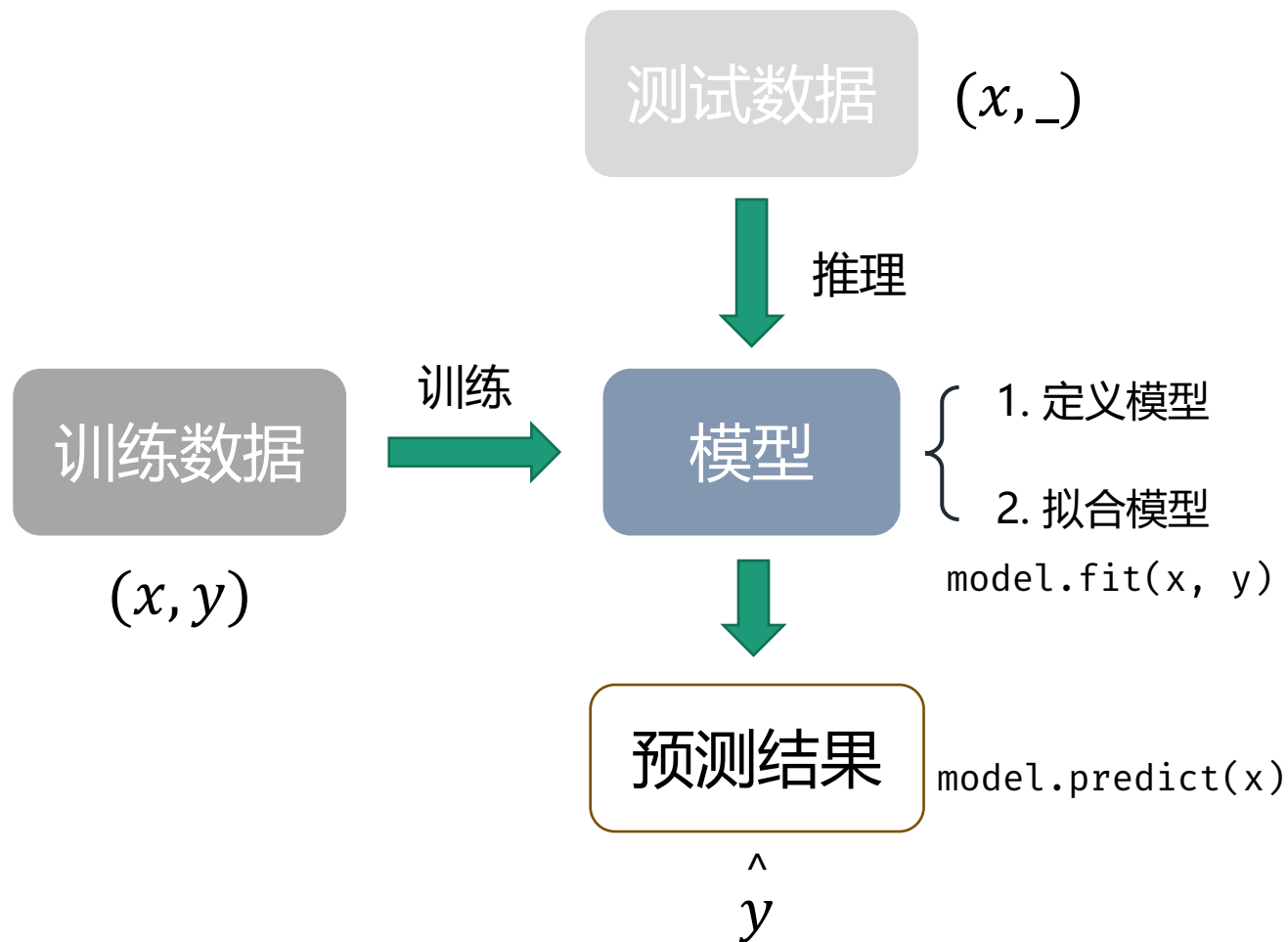
□ 环境安装

□ **使用示例**

□ 相关资源

使用示例

□ 一个典型的机器学习流程



□ 一个典型的机器学习步骤

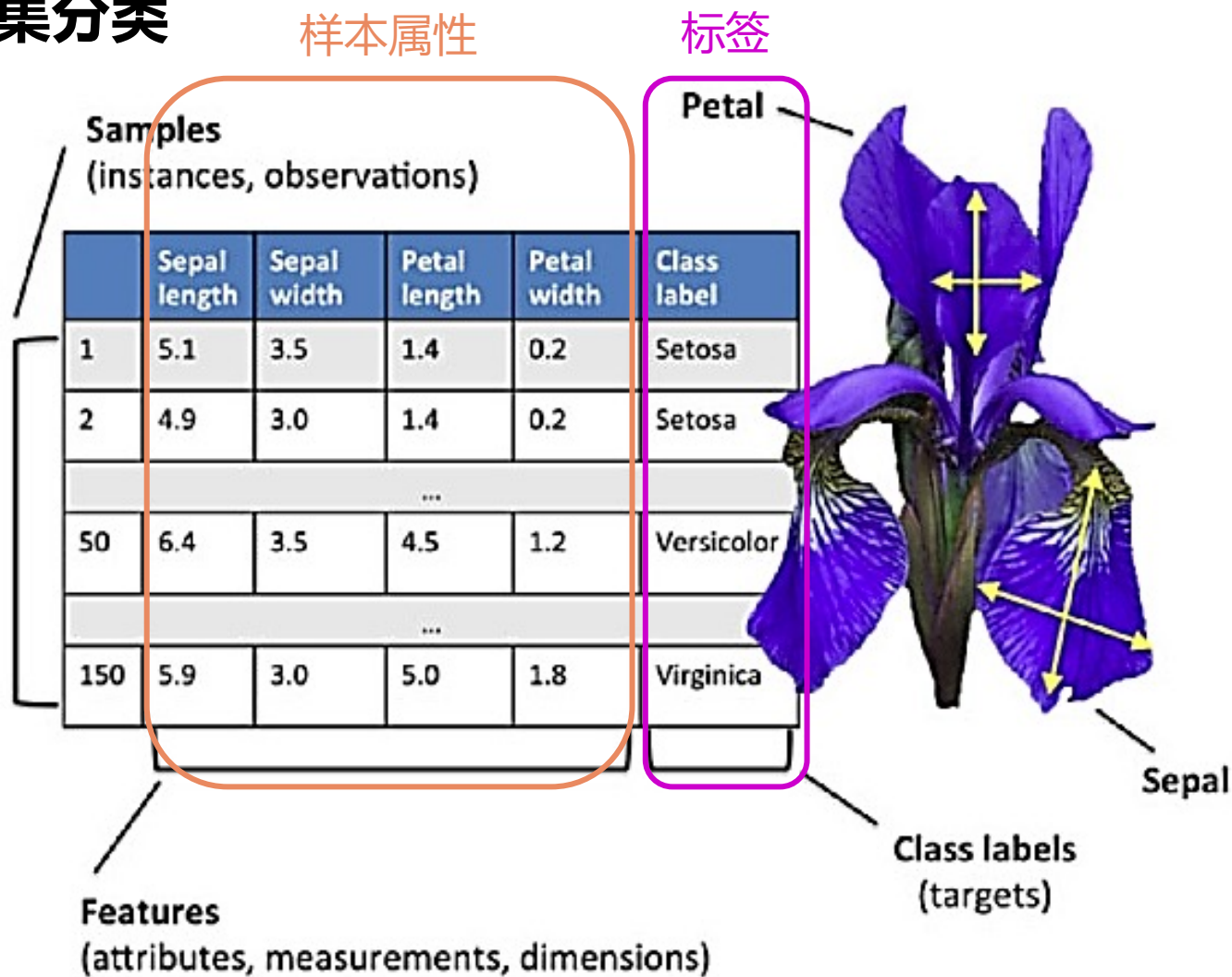
1. 导入或加载数据
2. 数据预处理
 - 归一化
 - 划分测试集、训练集
 - 特征选择
 - ...
3. 选择模型
 - 分类问题
 - 回归问题
 - ...
4. 训练模型
5. 模型评估

使用示例

□ 案例一：鸢尾花(iris)数据集分类

□ 数据集信息：

- 150个样本
- 每个样本包含四个维度的描述（属性）
- 3个类别标签



使用示例

□ 案例一：鸢尾花(iris)数据集分类



```
# 1. Import necessary libraries for machine learning tasks  
from sklearn.datasets import load_iris  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import confusion_matrix, accuracy_score  
import numpy as np
```

```
# 2. Loading the Iris dataset from sklearn  
iris = load_iris()  
X = iris.data      # Features  
y = iris.target    # Labels
```

使用示例

□ 案例一：鸢尾花(iris)数据集分类



```
# 3. Splitting the dataset into 80% training and 20% testing data  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
                                                    random_state=0)
```

```
# 4. Load model: initializing the KNN classifier with 5 neighbors  
knn = KNeighborsClassifier(n_neighbors=5)
```

```
# 5. Training the KNN model with the training data  
knn.fit(X_train, y_train)
```

使用示例



□ 案例一：鸢尾花(iris)数据集分类

```
# 6. Making predictions on the test dataset  
predictions = knn.predict(X_test)
```

```
# 7. Evaluating the model's performance using accuracy and confusion matrix  
print("Accuracy:", accuracy_score(y_test, predictions))  
print("Confusion Matrix:\n", confusion_matrix(y_test, predictions))
```

```
Accuracy: 0.9333333333333333  
Confusion Matrix:  
[[40  0  0]  
 [ 0 39  0]  
 [ 0  8 33]]
```

使用示例

□ 案例二：多项式曲线拟合（线性回归）

➤ **问题**：给定若干数据点 (x, y) ，满足关系 $y = 2x^2 + 4x + 1 + \epsilon$ ，其中 $\epsilon \sim \mathcal{N}(0,1)$
要求拟合一条二次曲线 $y = Ax^2 + Bx + C$

➤ **模型**：线性回归LinearRegression

➤ **数学表达式**： $y = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$

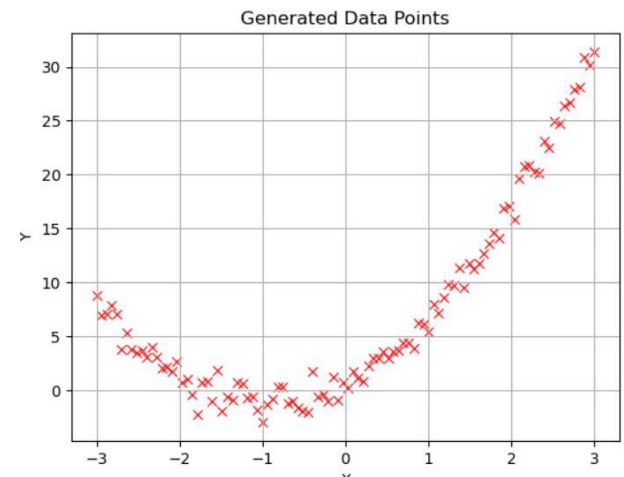
➤ **优化方法**：最小化
$$L(w) = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$
$$= \sum_{i=1}^N (y_i - (w_0 + w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in}))^2$$

使用示例

□ 案例二：多项式曲线拟合（线性回归）

```
# Import necessary package
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

```
# Define the polynomial function without noise
def polynomial(x):
    return 2 * x**2 + 4 * x + 1
# Define the noise function
def noise(n_samples, std=1):
    return np.random.normal(0, std, n_samples)
```



使用示例

□ 案例二：多项式曲线拟合（线性回归）

```
# Generate evenly distributed data within a specific range  
N_SAMPLE = 100  
x = np.linspace(-3, 3, N_SAMPLE)    # Generating 100 points from -3 to 3  
y = polynomial(x) + noise(N_SAMPLE) # Quadratic equation with some noise
```

```
# Reshaping x and y for sklearn  
x = x.reshape(-1, 1)  
y = y.reshape(-1, 1)
```


使用示例

□ 案例二：多项式曲线拟合（线性回归）

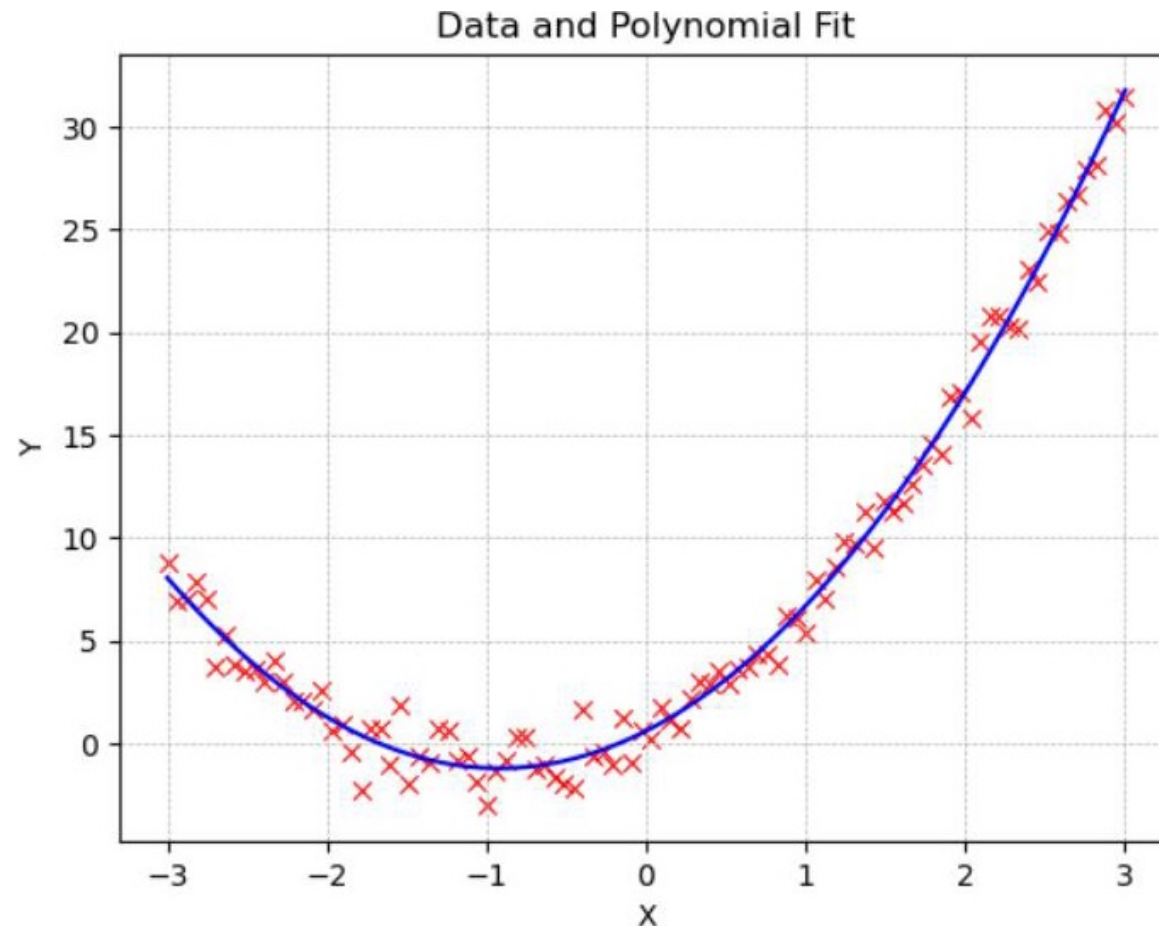
```
# Transforming features into polynomial features  
polynomial_features = PolynomialFeatures(degree=2)  
x_poly = polynomial_features.fit_transform(x)
```

```
# Training the Linear Regression model  
#  $y = A * x^2 + B * x + C$   
model = LinearRegression()  
model.fit(x_poly, y)  
y_poly_pred = model.predict(x_poly)
```

使用示例

□ 案例二：多项式曲线拟合（线性回归）

```
# Visualizing the original data  
points and the fitted polynomial  
curve  
plt.scatter(x, y, color='red',  
marker='x', linewidth=0.8)  
plt.plot(x, y_poly_pred,  
color='blue')  
plt.title('Data and Polynomial  
Fit')  
plt.xlabel('X')  
plt.ylabel('Y')  
plt.grid(True, which='both',  
linestyle='--', linewidth=0.5)  
plt.show()
```

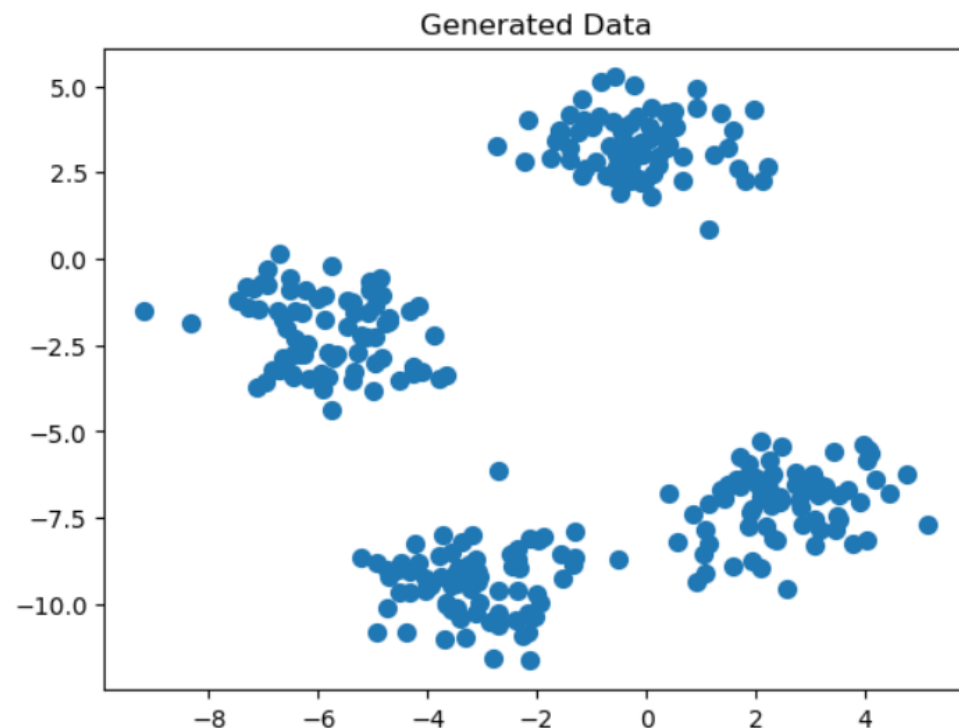


使用示例

□ 案例三：聚类

```
# Import libraries for clustering and data generation  
import matplotlib.pyplot as plt  
from sklearn.datasets import make_blobs  
from sklearn.cluster import KMeans
```

```
# Generate a synthetic dataset with 300 samples, 4 centers, and a standard deviation of 1.00  
X, y_true = make_blobs(n_samples=300,  
                        centers=4, cluster_std=1.00)  
# Plot the generated dataset  
plt.scatter(X[:, 0], X[:, 1], s=50)  
plt.title("Generated Data")  
plt.show()
```

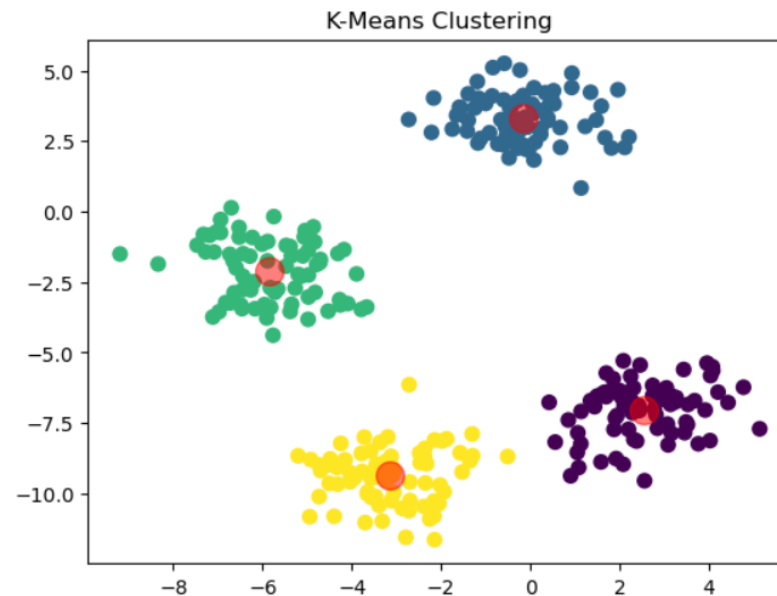


使用示例

□ 案例三：聚类

```
# Initialize KMeans with 4 clusters  
and fit it to the data  
kmeans = KMeans(n_clusters=4,  
n_init='auto')  
kmeans.fit(X)  
y_kmeans = kmeans.predict(X)
```

```
# Scatter plot of the data points with color coding for clusters  
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')  
# Plot the cluster centers in red  
centers = kmeans.cluster_centers_  
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5)  
plt.title("K-Means Clustering")  
plt.show()
```




□ Scikit-learn简介

□ 环境安装

□ 使用示例

□ 相关资源

相关资源

 [Install](#) [User Guide](#) [API](#) [Examples](#) [Community](#) [More ▾](#)

[Prev](#) [Up](#) [Next](#)

scikit-learn 1.4.1
[Other versions](#)

Please [cite us](#) if you use the software.

User Guide

- 1. Supervised learning
- 2. Unsupervised learning
- 3. Model selection and evaluation
- 4. Inspection
- 5. Visualizations
- 6. Dataset transformations
- 7. Dataset loading utilities
- 8. Computing with scikit-learn
- 9. Model persistence
- 10. Common pitfalls and recommended practices
- 11. Dispatching

User Guide

1. Supervised learning

- 1.1. Linear Models
- 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- 1.4. Support Vector Machines
- 1.5. Stochastic Gradient Descent
- 1.6. Nearest Neighbors
- 1.7. Gaussian Processes
- 1.8. Cross decomposition
- 1.9. Naive Bayes
- 1.10. Decision Trees
- 1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking
- 1.12. Multiclass and multioutput algorithms
- 1.13. Feature selection
- 1.14. Semi-supervised learning
- 1.15. Isotonic regression
- 1.16. Probability calibration
- 1.17. Neural network models (supervised)

2. Unsupervised learning

- 2.1. Gaussian mixture models

[Toggle Menu](#)

相关资源

- ❑ 官网地址: [scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation](https://scikit-learn.org/stable/index.html)
- ❑ 中文社区: [scikit-learn中文社区](#)
- ❑ Github Tutorial: [GitHub - jakevdp/sklearn_tutorial: Materials for my scikit-learn tutorial](https://github.com/jakevdp/sklearn_tutorial)
- ❑ vedio course: [Calmcode - scikit learn: Introduction](#)

谢谢!