

第四章 从问题求解到机器学习

主讲：陈建海

浙江大学计算机科学与技术学院
2024年9月



提纲

- 1 问题求解概述**
- 2 一般问题求解**
- 3 机器学习求解**
- 4 scikit-learn**



提纲

- 1 问题求解概述**
- 2 一般问题求解**
- 3 机器学习求解**
- 4 scikit-learn**

■问题求解是人工智能的一个分支。

- 人类的能力总是有限的，有许多复杂问题，人类难以甚至无法解决
- 于是人类发明了计算机来帮人类解决复杂问题，比如最优化决策的问题，装箱问题、倒水问题等

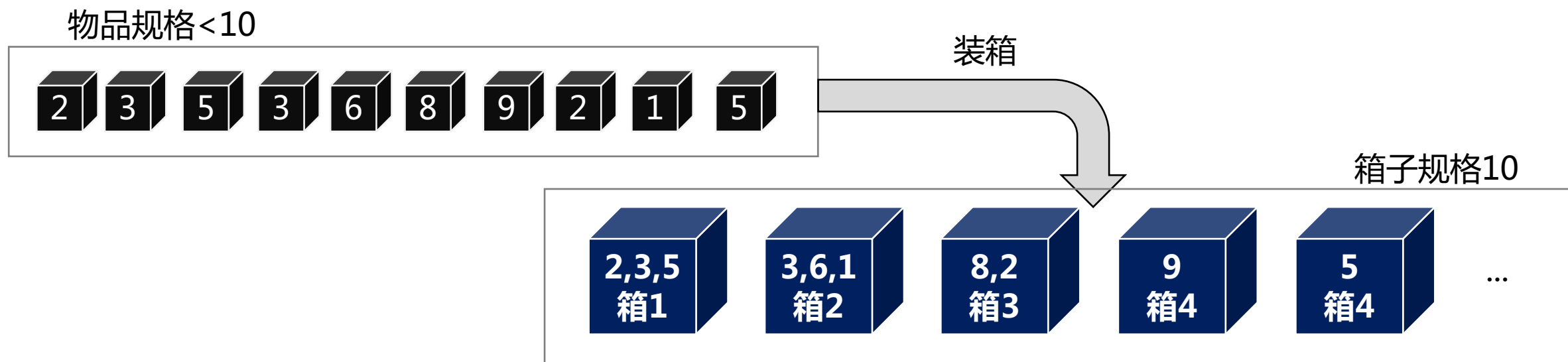
■如何求解？

- 通过计算机来解决，求解问题的方法是**设计算法**。

1 装箱问题

■问题描述：已知有N个物品（每个物品重量小于S）和一组箱子（箱子容量是S），要求寻找一种方法用最小个数的箱子把所有物品装完。

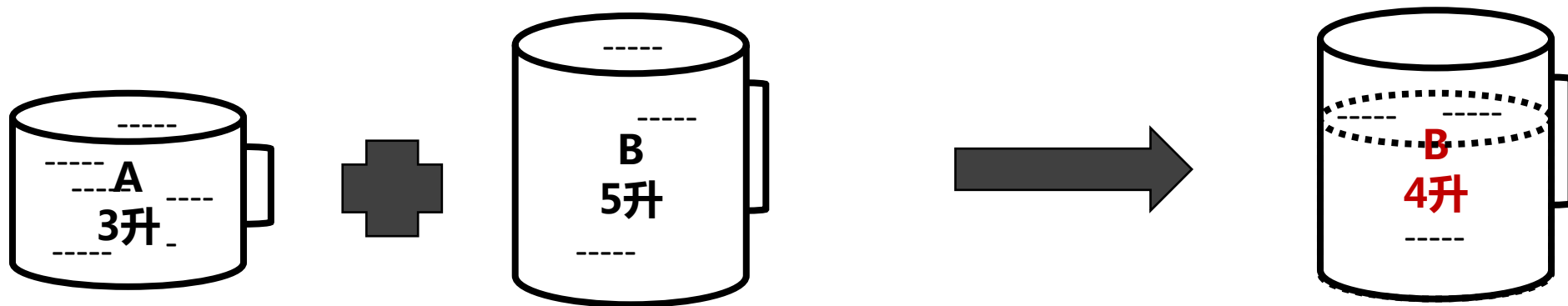
■例1：令 $S=10$ ，已知有10个物品（每个物品重量 ≤ 10 ），其重量分别是2、3、5、3、6、8、9、2、1、5，要求用最少的箱子数把物品装进箱子，箱子容量就是10。请问箱子数是多少。



1 倒水量水问题

■问题描述： 给定两个容量分别为 x 升和 y 升的水壶，以及一个目标 z 升的水量，我们需要找到一系列倒水操作，使得最终其中一个水壶中的水量恰好为 z 升，或者两个水壶中的水量之和为 z 升。

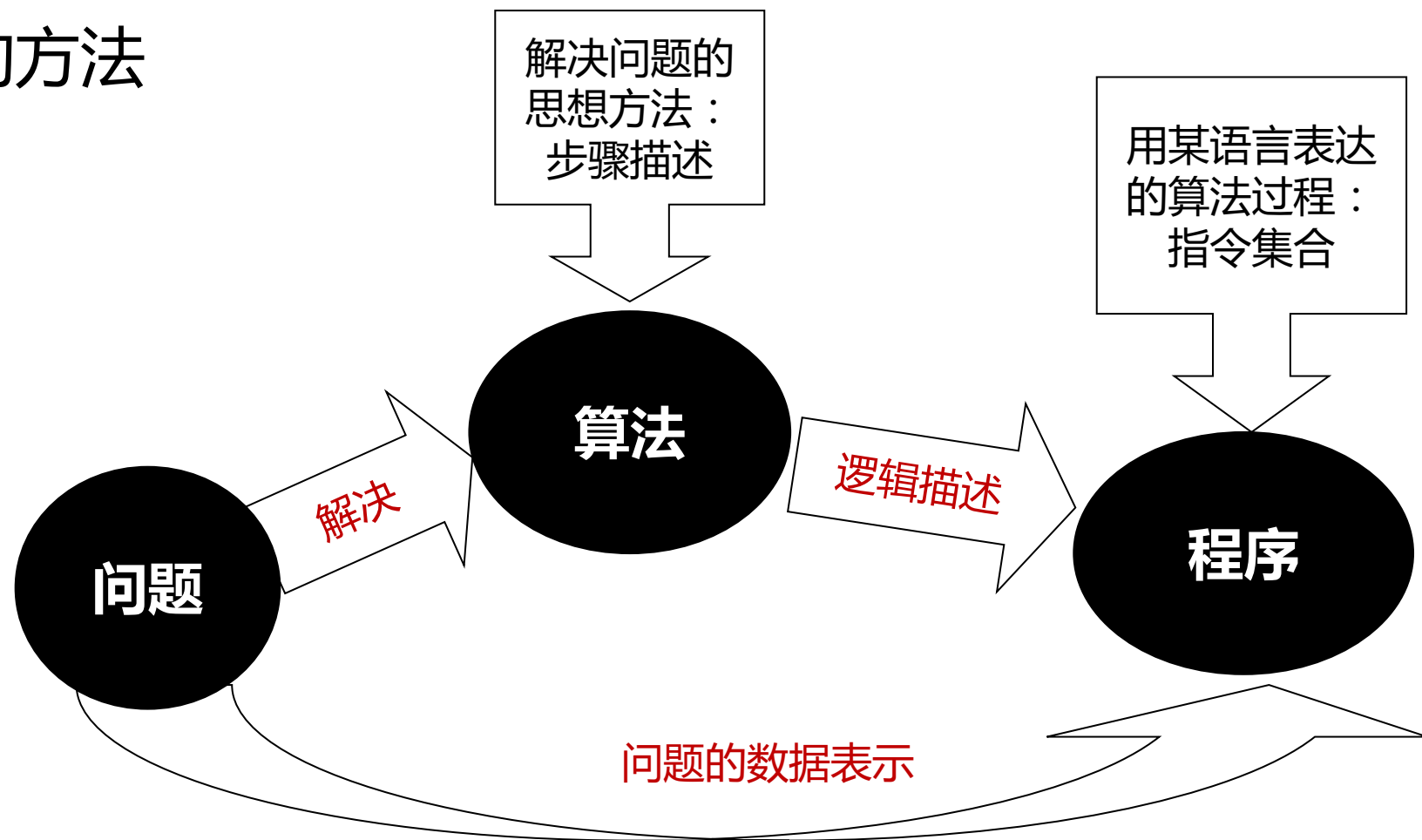
■例1: 令 $x=3$ ， $y=5$ ， $z=4$ ，即已知A、B两个水壶，A容量3升和B水壶容量5升，如何倒水测量出4升的水。



■问题求解的核心就是寻找解决方案并实现算法设计

■如何获得一个算法的方法

- (1) 贪心法
- (2) 分治法
- (3) 回溯法
- (4) 动态规划



1 贪心法

■ 基本思想

- 从小的方案推广到大的解决方法。**分阶段工作，在每一个阶段选择最好的方案，而不考虑其后的结果如何。**

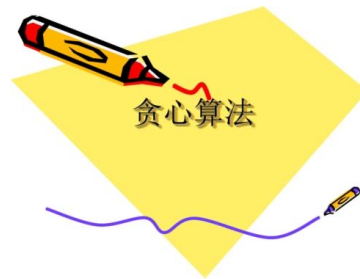
■ 贪心法主要用于**求解最优问题**，已发展成一种通用的算法设计技术。

■ 核心特性是：

- **可行性**——每一步选择必须满足问题的约束；
- **局部最优**——是当前可选择的方案中最优的；
- **不可撤销性**——选择一旦做出，在算法的其后步骤中不能被撤消。

贪心法的优缺点

- 贪心法不能确定最后解是最优的，也不能用于求解最大或最小问题。
- 在算法的效率上，贪心法快速，程序实现需要的内存开销也较小。
- 但遗憾的是，它们往往是不正确的。
- 然而一旦被证明是正确的，其执行效率和速度有很大的优势。



1 贪心法—举例—找零钱问题

■假设只有 1 分、2 分、五分、1 角、二角、五角、1 元的硬币。在超市结账时，如果需要找零钱，收银员希望将最少的硬币数找给顾客。那么，给定需要找的零钱数目，如何求得最少的硬币数呢？

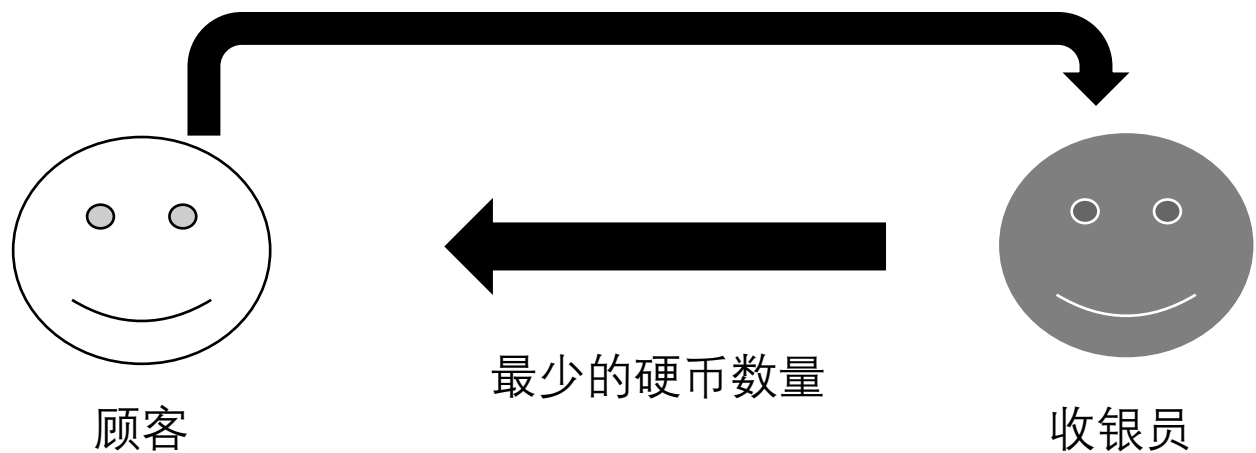
- 贪心的方法

先用最大面值换

再用次大面值换

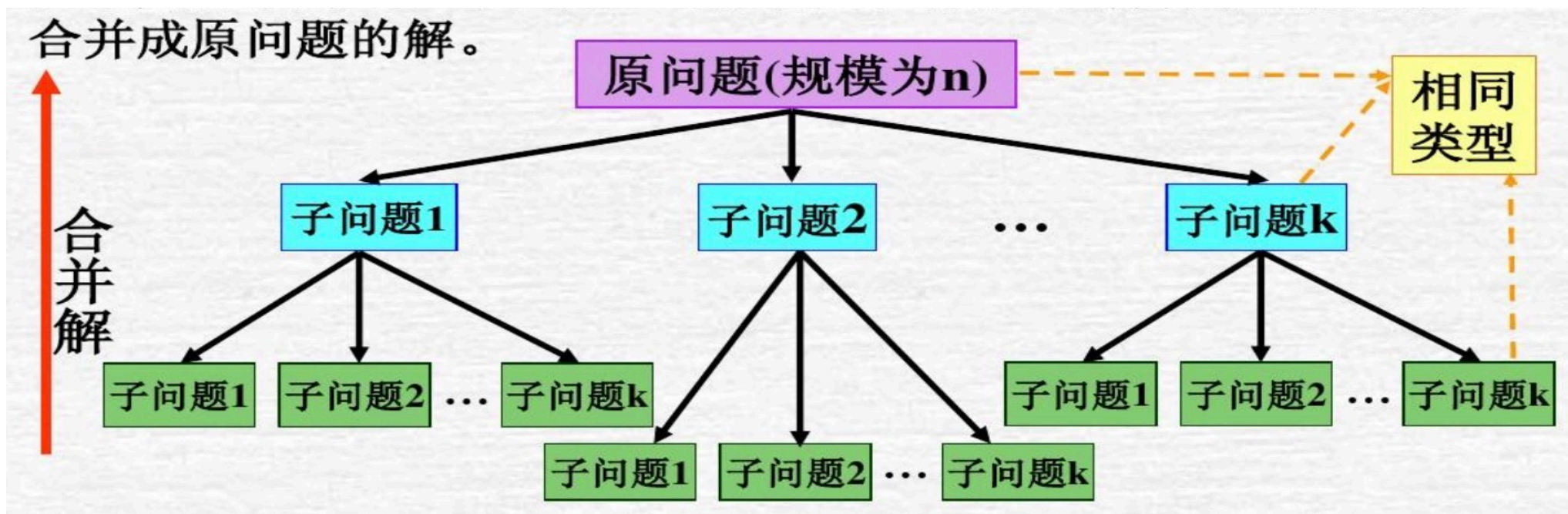
.....

最后用最小面值换



1 分治法

- 基本思想：分而治之，将一个较大规模的问题分解为若干个较小规模的子问题，找出子问题的解，然后把各个子问题的解合并成整个问题的解。
- 分治法的分（Divide）：指划分较大问题为若干个较小问题，递归求解子问题；
- 分治法的治（Conquer）：指从小问题的解构建大问题的解。



1 分治法举例—金块问题求解

- 金块问题：（ 挑出最重和最轻的两个金块 ）
- $n=8$ 时，普通方法需要的比较次数： $(n-1)+(n-2) = 13$ 。
- 分治法的比较次数：

8								比较次数
4				4				
2		2		2		2		
L	H	L	H	L	H	L	H	4
	L	H			L	H		4
		L	H					2

10

10

- $n=16$ 时，普通方法需要的比较次数： $(n-1)+(n-2) = 29$ 。
- 分治法的比较次数： $10+10+2 = 22$
- $C(n) = 2 C(n/2) + 2$ (当 $n>2$) = $3n/2 - 2$ (当 n 为2的幂)

金块问题程序代码：

```
def min_max(a):
    if len(a)==1:
        return (a[0],a[0])
    elif len(a)==2:
        return (min(a),max(a))
    else:
        m=len(a)//2
        lmin,lmax=min_max(a[:m])
        rmin,rmax=min_max(a[m:])
        return (min(lmin,rmin),max(lmax,rmax))
```

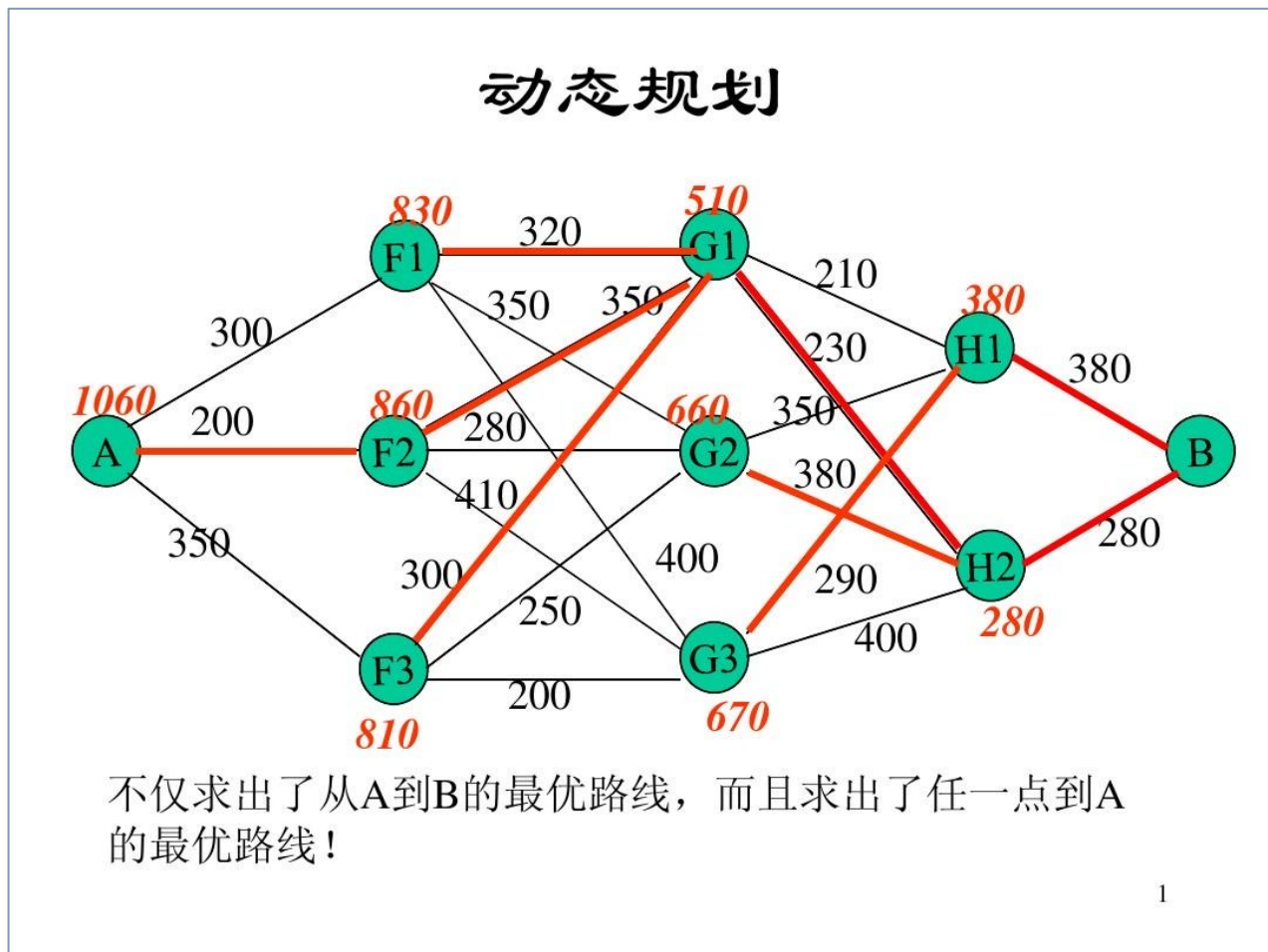
```
a,b=min_max([3,8,9,4,10,5,1,17])
print("最小值是",a,"， 最大值是",b)
```

1 动态规划法

■动态规划的基本思想：

- 如果一个较大问题可以被分解为若干个子问题，且子问题具有重叠，可以将每个子问题的解存放到一个表中，这样就可以通过**查表**解决问题。

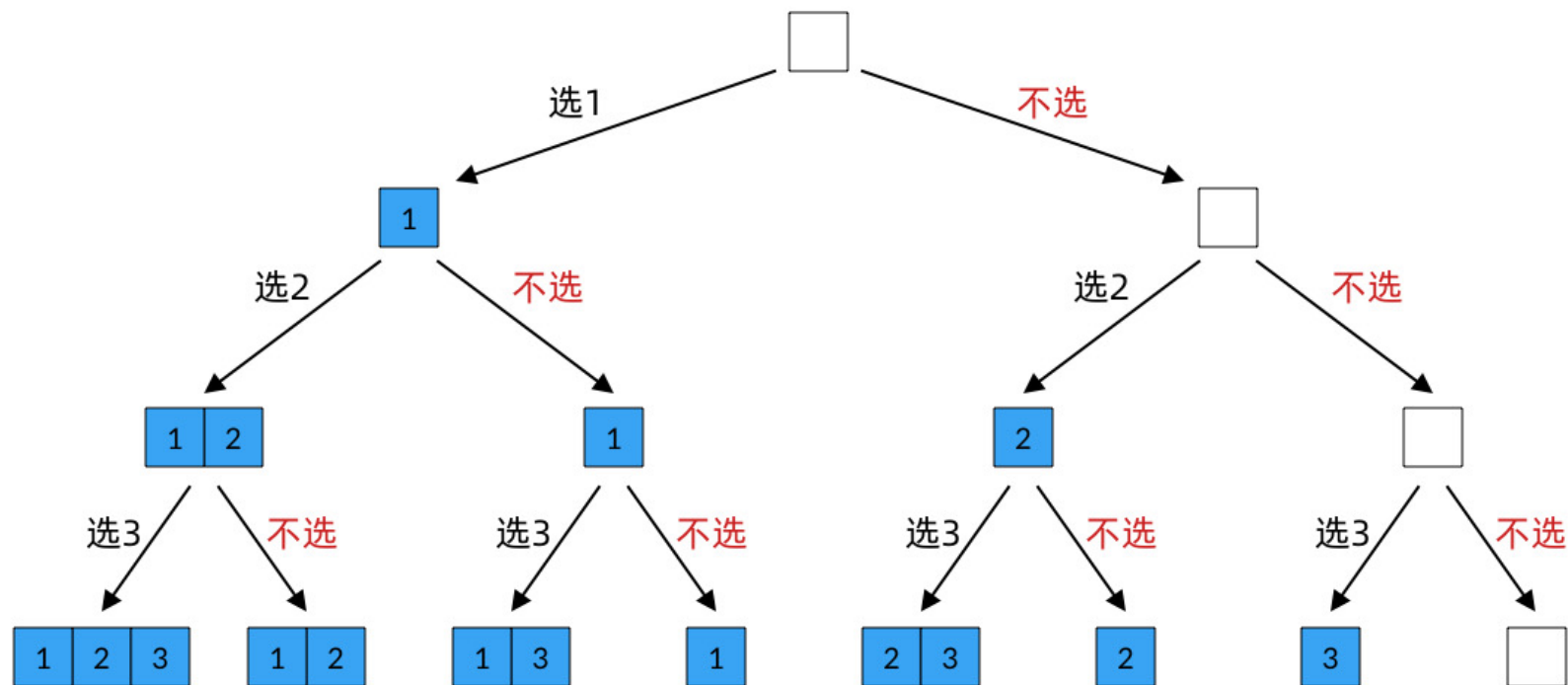
■核心思想是以空间换时间！



1

回溯法

- 回溯法也叫**穷尽搜索法**（Brute-Force Search），尝试分步地去解决问题。
- **基本思想**：在分步解决问题的过程中，当它通过尝试发现现有的分步答案不能得到**有效的、正确的**解答的时候，将**取消上一步甚至上几步**的计算，再**通过其他的可能的分步解答**再次尝试寻找问题的答案。
- 通常使用**递归**实现。



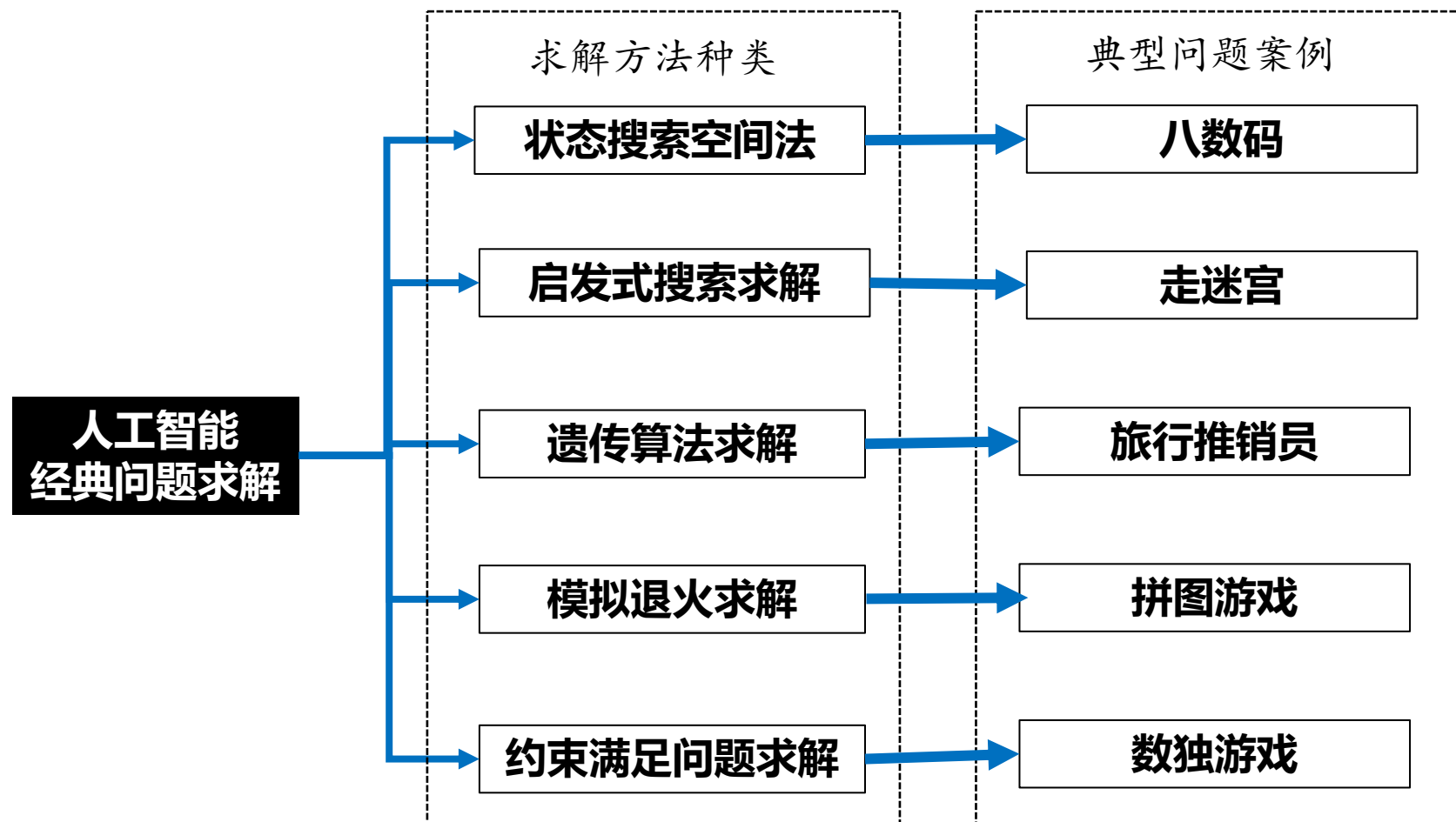


提纲

- 1** 问题求解概述
- 2** 一般问题求解
- 3** 机器学习求解
- 4** scikit-learn

2 人工智能经典问题求解

- 人工智能解决的是复杂问题，人类难以解决的问题，需要借助计算机算法工具来辅助解决的问题。



2 八数码与状态搜索空间法

■八数码：在一个3*3的网格中随机放置了1-8八个数码。其中有一个网格是空着的。这个空网格可以跟上下左右四个方向的任何一个临近的数码交换。但不能跟斜方向上的数码交换。

比如上图中空网格可以和右边的那个数码3相交换，得到的子状态如图2（b）。

八数码问题就是研究如何用最少的次数移动空网格，从而使得八个数码最终呈现出如下所示的终止状态，如图2（c）。

6	2	5
	3	8
4	7	1

(a) 起始状态

6	2	5
3		8
4	7	1

(b) 子状态

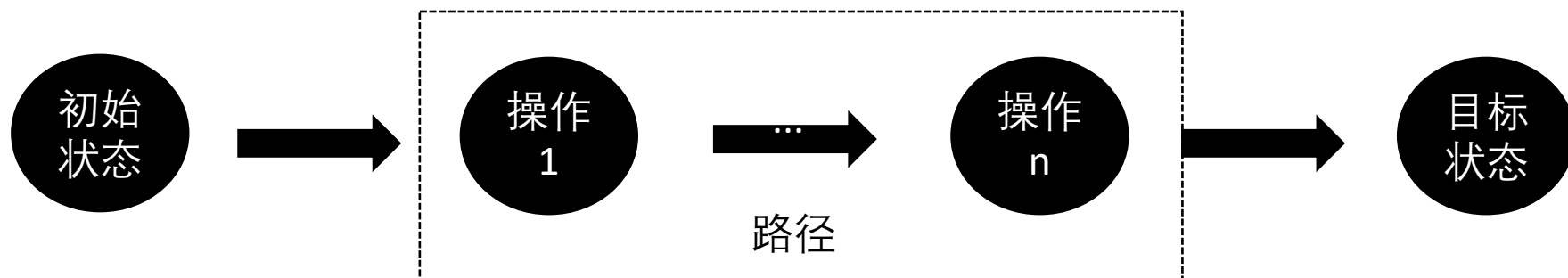
1	2	3
8		4
7	6	5

(c) 终止状态

3 八数码与状态搜索空间法

■如果一个问题可以被定义为状态空间中的一个初始状态，通过一系列操作或动作转换直到到达目标状态，这样的问题就是状态空间（State Space）问题。状态空间问题包含四个要素：

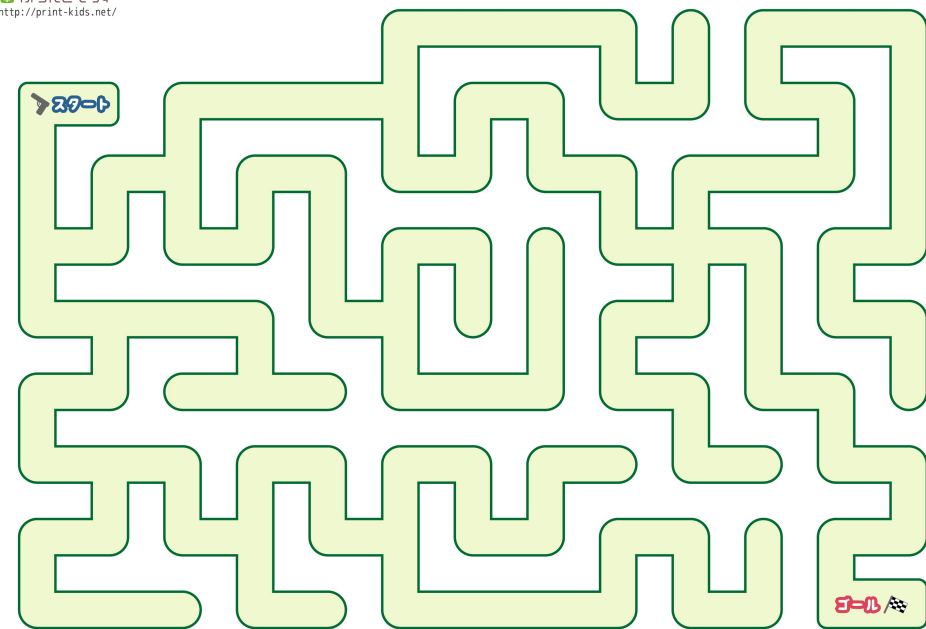
- （1）初始状态（Initial State）：问题开始时的状态。
- （2）目标状态（Goal State）：问题希望达到的状态。
- （3）操作（Operators）：从一个状态转换到另一个状态的规则或操作。
- （4）路径（Path）：从初始状态到目标状态的一系列操作的集合。



4 走迷宫与启发式搜索

■走迷宫问题就是一种启发式搜索问题。想象一个简单的迷宫问题，你需要从起点到达终点。启发式搜索可以用距离终点的直线距离作为启发式函数。算法在每一步都会选择那些看起来离终点更近的路径。通过这种方式，即使不是每一步都朝着最终目标直行，搜索过程也会倾向于向终点方向前进。

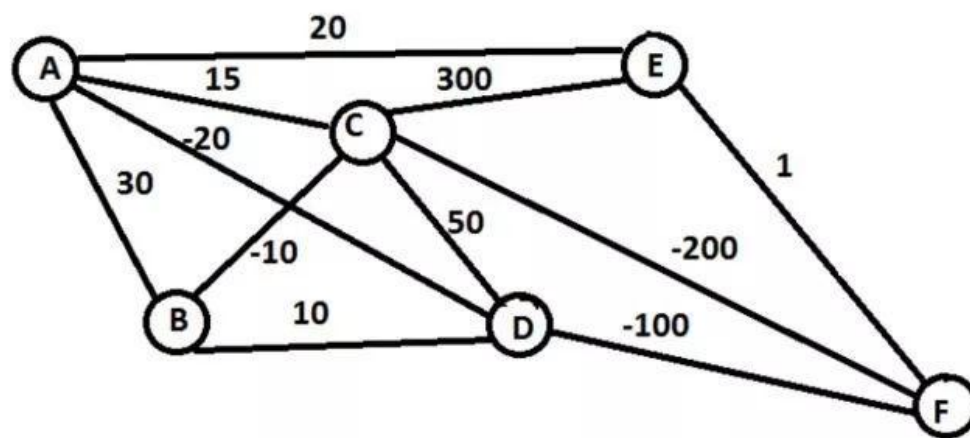
プリントマップ
<http://print-kids.net/>



■启发式搜索是在搜索空间中查找问题解决方案的技术，它使用一个启发式函数来评价选择哪个分支可能会导向最佳解。启发式函数可以看作是一种“直觉”或“猜想”，帮助算法决定下一步最有希望的路径。

4 旅行推销员与遗传算法求解

- 遗传算法是一种模拟自然选择的优化技术，它们使用生物进化的概念，如变异、交叉和选择来解决问题。



- 典型例子就是旅行推销员问题。
- 旅行推销员问题（TSP）要求找到最短的路径，访问一系列城市并返回原点。使用遗传算法，可以随机生成多个“种群”，每个种群是一种可能的路径。
- 通过交叉和变异产生新的路径，并使用适应度函数（在这个例子中是总距离）来选择最佳路径，随着时间的推移，这个过程可能会找到一个非常接近最短可能路径的解决方案。

4 拼图与模拟退火

- 模拟退火是一种概率性的技术，灵感来源于金属退火过程。它在搜索过程中引入随机性，允许算法在早期阶段进行“坏”的移动，以避免陷入局部最优解。
- 典型的例子就是拼图游戏。
- 考虑一个拼图游戏，你需要将一组零件放置在正确的位置。



- 模拟退火会从一个随机的拼图配置开始，然后在每一步尝试移动一块拼图。即使某些移动一开始看起来并不朝向解决方案，算法也会接受它们，这样做的目的是为了探索更大的搜索空间。随着时间的推进，算法变得越来越不可能接受那些使拼图状态变差的移动，最终趋向于解决方案。

2 数独与约束满足问题求解

■数独是一个经典的约束满足问题（CSP），其中每个数字必须在一个3x3的子网格中唯一出现，并且在每行和每列都唯一。求解数独的算法需要找到一种数字的分配方式，满足所有行、列以及子网格都没有重复数字。



- 约束满足问题是一种问题，其中变量需要满足一定的约束条件。
- CSP求解器的目的是找到满足所有约束的变量的赋值。

数独



提纲

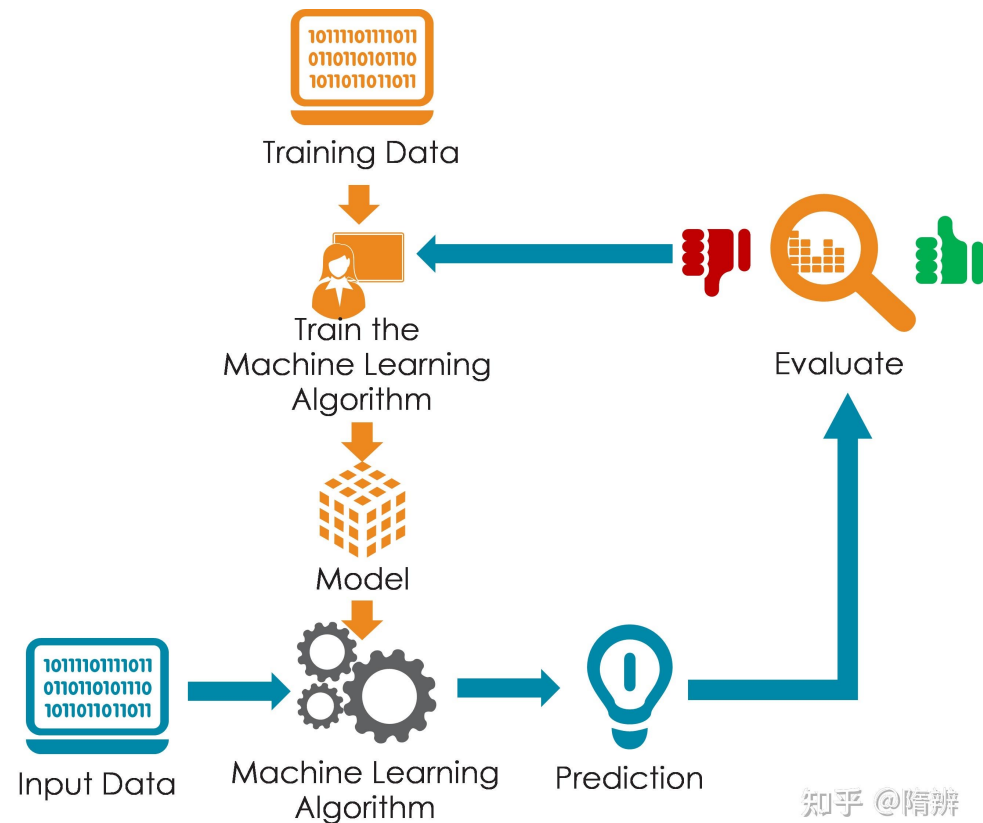
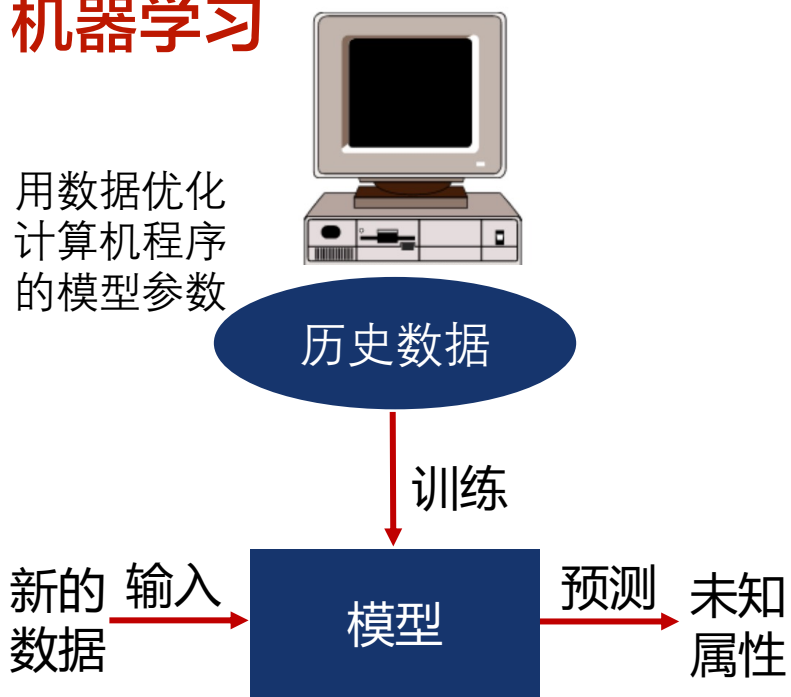
- 1** 问题求解
- 2** 一般问题求解
- 3** 机器学习求解
- 4** scikit-learn

3 什么是机器学习

- 机器学习是AI的核心领域之一，是一种基于数据统计建模的问题求解方法。
- 采用AI解决任何一个任务往往都需要的步骤：

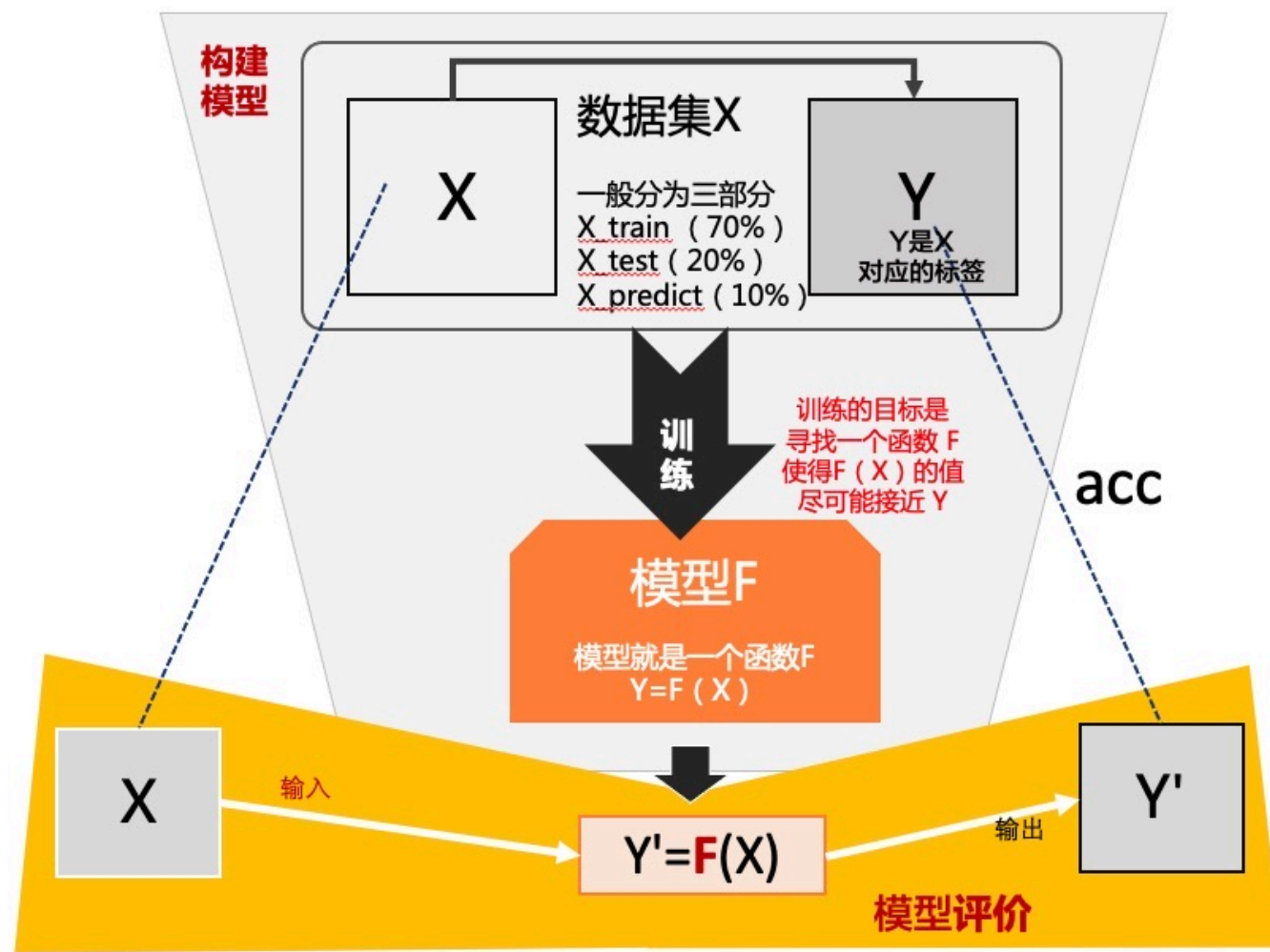
- 定义模型
- 训练模型
- 测试模型
- 评估模型

机器学习



3 机器学习的定义

- 已知给出一个数据集 X , Y , Y 与 X 存在某种对应关系, Y 称为 X 的标签。
- 现在要寻找一个函数或模型 F , 使得 F 作用在 X 上之后的结果是 $Y' = F(X)$, 满足 $|Y - Y'|$ 尽可能小。
- 我们把利用机器的算法寻找函数 F 的过程称为机器学习。



3 机器学习的类型

有监督学习

- 若所有的 X 都有 Y 对应，则是有监督学习。
- 就是有老师带着你学习。

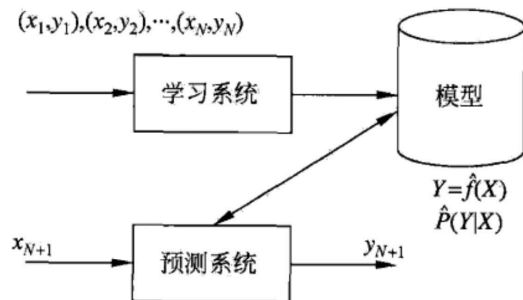


图 1.1 监督学习问题

无监督学习

- 若没有 Y ，学习 X 内部的知识，则称为无监督学习。
- 没有老师带着你，你自己学习。

半监督学习

- 若一部分 X 有 Y ，则称为半监督学习。
- 老师教了你部分的答案，其余不告诉你，这样的学习就是半监督的。

3 机器学习五个要素

■ 机器学习求解方法包含五个要素：

■ 数据

- 结构化的（如表格数据）或非结构化的（如文本、图像、音频）

■ 模型

- 用于进行预测或分类的数学表示，模型通过从数据中学习并调整其参数来提高性能。

■ 训练：

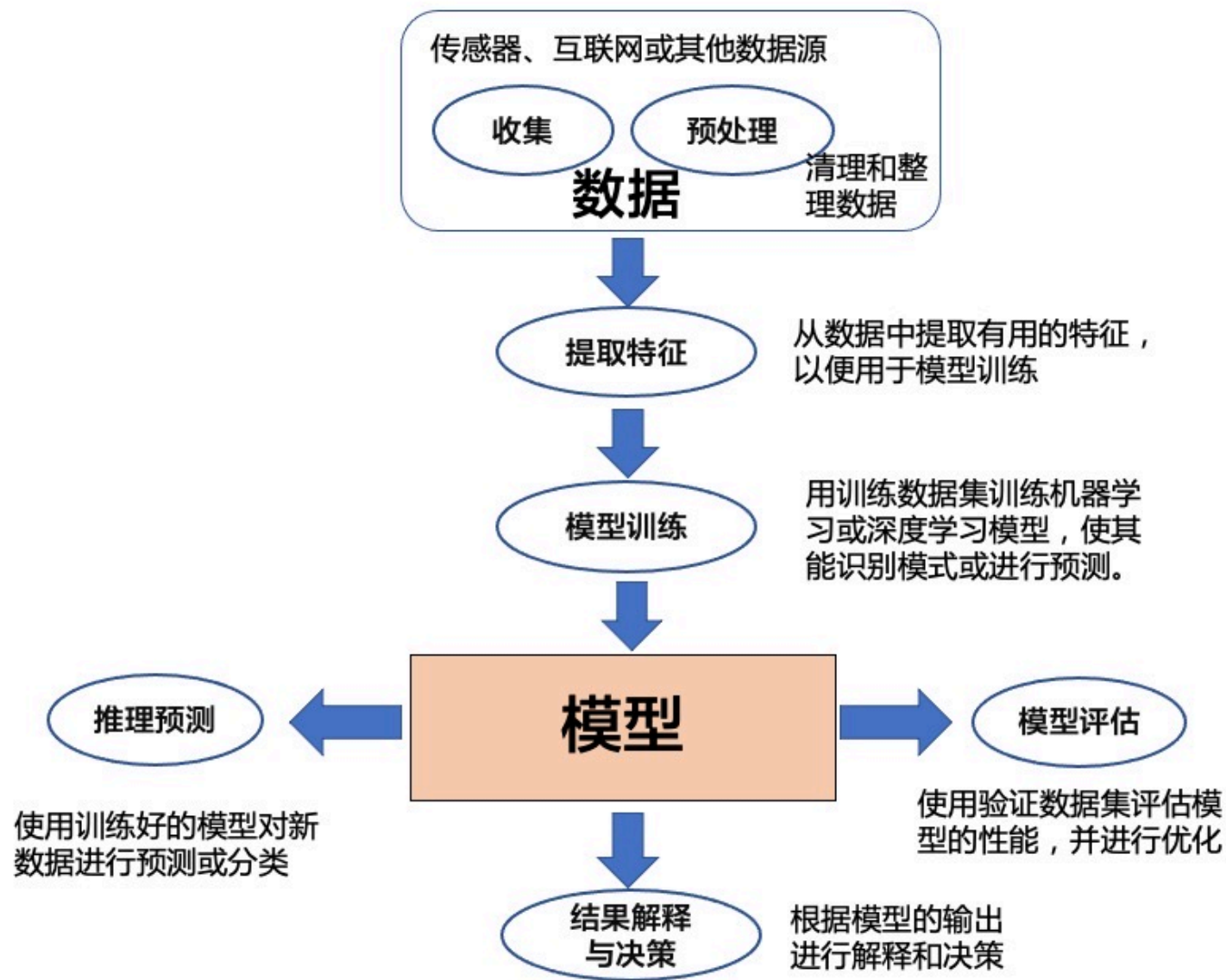
- 训练过程是模型从数据中学习的过程，通常涉及最小化某个损失函数以优化模型参数。

■ 预测：

- 基于模型可以对新数据进行预测或分类

■ 评估：

- 各种评估指标（如准确率、精确率、召回率、F1分数等）来衡量模型的性能



3 机器学习的工具和框架

■ 机器学习的实现方法

- 回归分类方法
- 神经网络方法

■ 机器学习工具

- 机器学习库: scikit-learn

■ 深度学习框架

- TensorFlow
- PyTorch

■ 高级工具

- keras: 高级神经网络API
- XGBoost: 优化的分布式梯度提升库

3 机器学习的挑战问题

- （1）数据质量问题，数据的准确性、完整性和一致性直接影响模型的性能。
- （2）过拟合问题。模型在训练数据上表现很好，但在新数据上表现不佳，通常是由于模型过于复杂导致的。
- （3）计算资源问题。训练复杂模型需要大量的计算资源和时间。
- （4）模型解释性问题。一些复杂模型（如深度神经网络）难以解释其决策过程，这在某些应用场景中可能是一个问题。



提纲

- 1 问题求解**
- 2 一般问题求解**
- 3 机器学习求解**
- 4 scikit-learn**

4 scikit-learn总体介绍

□ Scikit-learn是Python中最受欢迎的机器学习库，提供了

- ✓ 丰富多样的机器学习算法：从简单的线性回归到复杂的深度学习模型。
- ✓ 丰富的实用工具：帮助你进行特征工程、数据预处理、模型选择等。
- ✓ 完善的文档，有大量的示例代码和教程，上手简单。

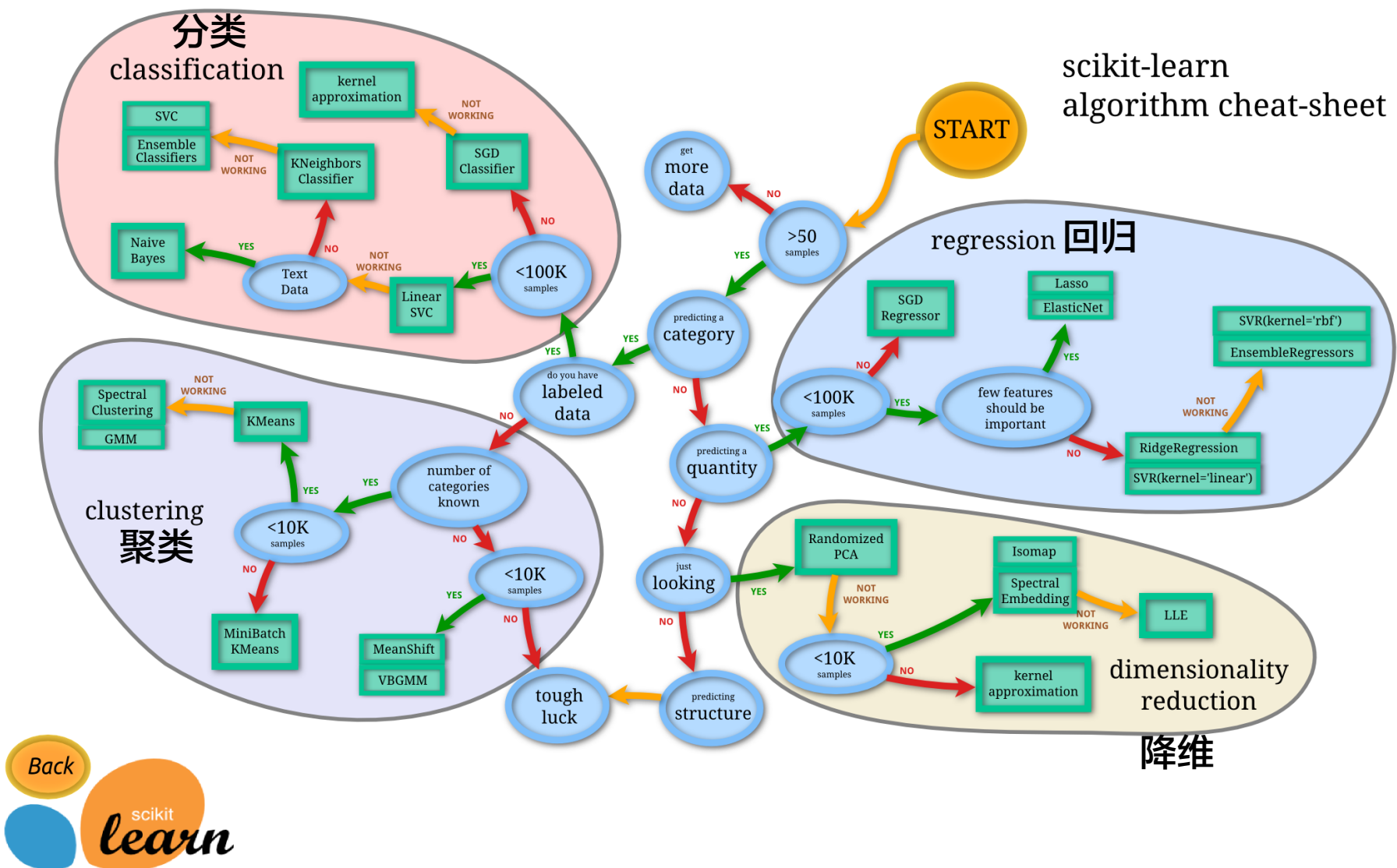
□ 中文社区：<https://scikit-learn.org.cn/>

□ Sklearn使用，需要熟悉的基本操作，包括

- 如何加载数据，选择模型，训练和评估模型，以及调整模型的参数以获得更好的性能等。

4 scikit-learn算法组成

□由图中，可以看到机器学习 sklearn 库的算法主要有四类：分类，回归，聚类，降维。



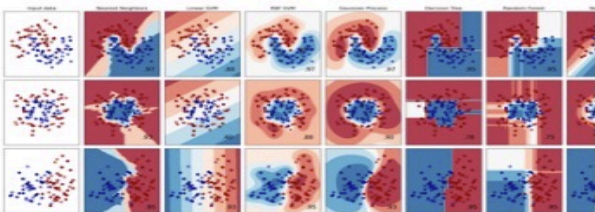
4 scikit-learn 算法

分类

标识对象所属的类别。

应用范围：垃圾邮件检测，图像识别。

算法：SVM 最近邻 随机森林 更多...



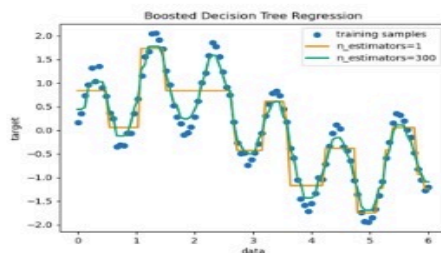
Examples

回归

预测与对象关联的连续值属性。

应用范围：药物反应，股票价格。

算法：SVR 最近邻 随机森林 更多...



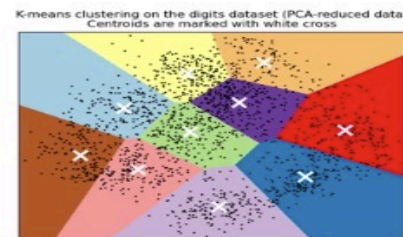
Examples

聚类

自动将相似对象归为一组。

应用：客户细分，分组实验成果。

算法：K-均值 谱聚类 MeanShift 更多...



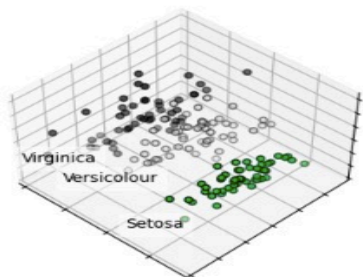
Examples

降维

减少要考虑的随机变量的数量。

应用：可视化，提高效率。

算法：K-均值 特征选择 非负矩阵分解 更多...



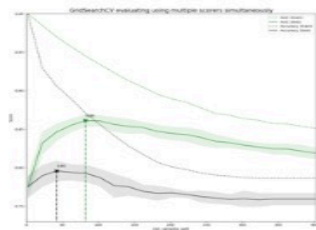
Examples

模型选择

比较，验证和选择参数和模型。

应用：通过参数调整改进精度。

算法：网格搜索 交叉验证 指标 更多...



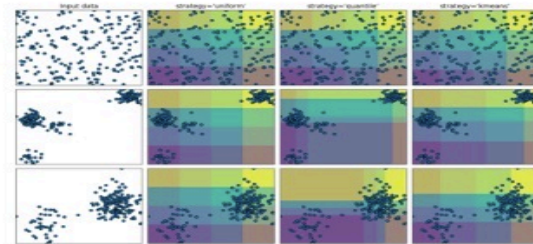
Examples

预处理

特征提取和归一化。

应用程序：转换输入数据，例如文本，以供机器学习算法使用。

算法：预处理 特征提取 更多...



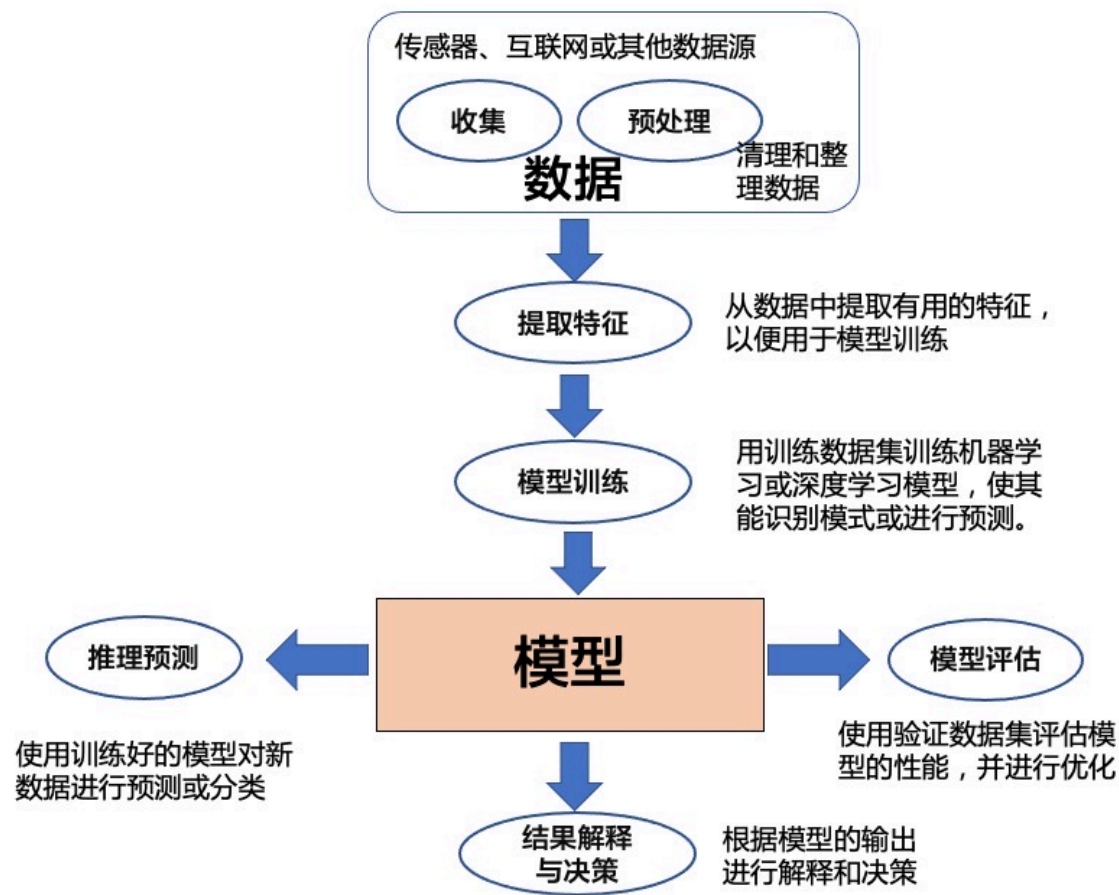
Examples

□1、Sklearn的下载安装

- 安装好python环境
- 输入 `python -m pip install scikit-learn` 或者
- 输入 `python -m pip install scikit-learn -i https://pypi.tuna.tsinghua.edu.cn/simple`

4 scikit-learn使用的基本操作

- ✓ (1) 准备、加载数据和预处理 (特征工程)
- ✓ (2) 选择模型
- ✓ (3) 训练和评估模型
- ✓ (4) 调整模型的参数以获得更好的性能



4 数据集

□sklearn常见数据集

□sklearn.datasets.load_*

获取小规模数据集，数据包含在 datasets 里

□sklearn.datasets.fetch_*(data_home=None)

✓ 获取大规模数据集，需要从网络上下载，函数的第一个参数是 data_home，表示数据集下载的目录，默认是 /scikit_learn_data/

举例：获取鸢尾花数据集

	数据集名称	调用方式	适用算法	数据规模
小数据集	波士顿房价	load_boston()	回归	506*13
小数据集	鸢尾花数据集	load_iris()	分类	150*4
小数据集	糖尿病数据集	load_diabetes()	回归	442*10
大数据集	手写数字数据集	load_digits()	分类	5620*64
大数据集	Olivetti脸部图像数据集	fetch_olivetti_faces	降维	400*64*64
大数据集	新闻分类数据集	fetch_20newsgroups()	分类	-
大数据集	带标签的人脸数据集	fetch_lfw_people()	分类、降维	-
大数据集	路透社新闻语料数据集	fetch_rcv1()	分类	804414*47236

```
from sklearn.datasets import load_iris
# 获取鸢尾花数据集
iris = load_iris()
print("鸢尾花数据集的返回值: \n", iris.keys())
```

鸢尾花数据集的返回值:

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```


4 数据集

□sklearn可以自己创建数据集，sklearn中的samples generator包含的大量创建样本数据的方法。

```
datasets.make_blobs ([n_samples, n_features, ...])
```

```
datasets.make_classification ([n_samples, ...])
```

```
datasets.make_circles ([n_samples, shuffle, ...])
```

```
datasets.make_friedman1 ([n_samples, ...])
```

```
datasets.make_friedman2 ([n_samples, noise, ...])
```

```
datasets.make_friedman3 ([n_samples, noise, ...])
```

```
datasets.make_gaussian_quantiles ([mean, ...])
```

```
datasets.make_hastie_10_2 ([n_samples, ...])
```

```
datasets.make_low_rank_matrix ([n_samples, ...])
```

```
datasets.make_moons ([n_samples, shuffle, ...])
```

```
datasets.make_multilabel_classification ([...])
```

```
datasets.make_regression ([n_samples, ...])
```

```
datasets.make_s_curve ([n_samples, noise, ...])
```

```
# 导入内建数据集
from sklearn.datasets import load_iris

# 获取鸢尾花数据集
iris = load_iris()

# 获得 ndarray 格式的变量 X 和标签 y
X = iris.data
y = iris.target

# 获得数据维度
n_samples, n_features = iris.data.shape
print(n_samples, n_features)
```

□ **预处理**是通过一些**转换函数**将**特征数据转换成更加适合算法模型的特征数据**过程。

□ 常见方法

- 数据标准化
- 数据二值化
- 标签编码
- 独热编码
- 等等。

□数据标准化和归一化是将数据映射到一个小的浮点数范围内，以便模型能快速收敛。标准化有多种方式。

1) min-max标准化 (对象名为MinMaxScaler)，该方法使数据落到[0,1]区间：

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2) Z-score标准化 (对象名为StandardScaler)，该方法使数据满足标准正态分布：

$$x' = \frac{x - \bar{X}}{S}$$

3) 归一化 (对象名为Normalizer，默认为L2归一化)：

$$x' = \frac{x}{\sqrt{\sum_j^m x_j^2}}$$

```
# min-max标准化
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()
sc.fit(X)
results = sc.transform(X)
print("放缩前:", X[1])
print("放缩后:", results[1])
```

```
放缩前: [4.9 3.  1.4 0.2]
放缩后: [0.16666667 0.41666667 0.06779661 0.04166667]
```

```
# Z-score标准化
from sklearn.preprocessing import StandardScaler

#将fit和transform组合执行
results = StandardScaler().fit_transform(X)

print("放缩前:", X[1])
print("放缩后:", results[1])
```

```
放缩前: [4.9 3.  1.4 0.2]
放缩后: [-1.14301691 -0.13197948 -1.34022653 -1.3154443 ]
```

```
# 归一化
from sklearn.preprocessing import Normalizer

results = Normalizer().fit_transform(X)

print("放缩前:", X[1])
print("放缩后:", results[1])
```

```
放缩前: [4.9 3.  1.4 0.2]
放缩后: [0.82813287 0.50702013 0.23660939 0.03380134]
```

4 数据二值化

- 使用**阈值过滤器**将数据转化为**布尔值**，即为二值化。
- 使用**Binarizer**对象实现数据的二值化。

```
# 二值化, 阈值设置为3
from sklearn.preprocessing import Binarizer

results = Binarizer(threshold=3).fit_transform(X)

print("处理前:", X[1])
print("处理后:", results[1])
```

处理前: [4.9 3. 1.4 0.2]

处理后: [1. 0. 0. 0.]

4 标签编码和独热编码

■**标签编码**：使用 **LabelEncoder** 将不连续的**数值或文本变量**转化为有序的数值型变量。

标签编码

```
from sklearn.preprocessing import LabelEncoder
LabelEncoder().fit_transform(['apple', 'pear', 'orange', 'banana'])
```

```
array([0, 3, 2, 1])
```

- **独热编码**：对于**无序的离散型**特征，其数值大小并没有意义，可进行**one-hot编码**，将其特征的m个可能值转化为**m个二值化（0、1）**特征。

■利用 OneHotEncoder 对象实现。

独热编码

```
from sklearn.preprocessing import OneHotEncoder
```

```
results = OneHotEncoder().fit_transform(y.reshape(-1,1)).toarray()
```

```
print("处理前: ", y)
```

```
print("处理后: ", results[1])
```

[illegible]

处理后: [1. 0. 0.]

4 mo sklearn学习

<http://mo.zju.edu.cn>

总结

1 问题求解概述

2 传统问题求解

3 机器学习求解

4 scikit-learn

Q&A
谢谢各位



- 主 讲 人：陈建海
- 联系方式：13958011808
- <http://person.zju.edu.cn/cjh>