

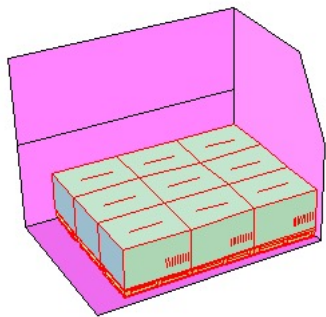
# 第五章 机器学习与监督学习

**主讲：陈建海**

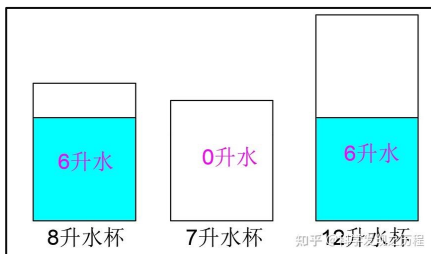
浙江大学计算机科学与技术学院  
2024年9月

## 经典复杂问题求解

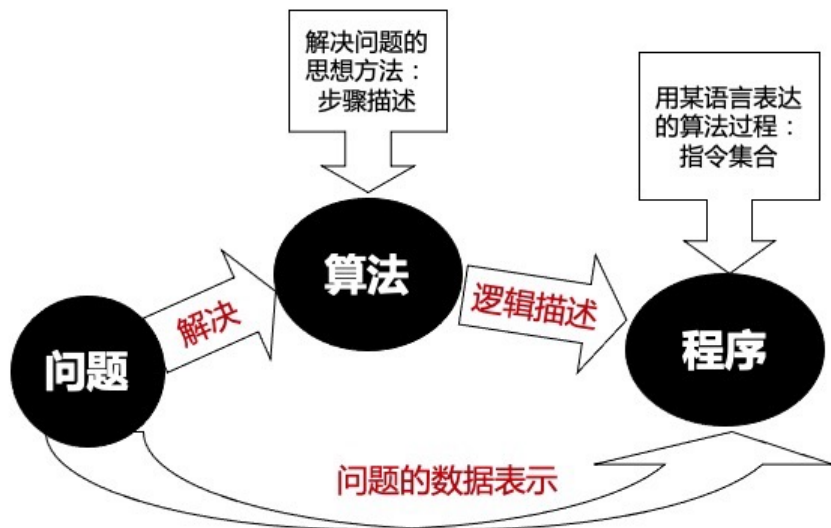
装箱问题



倒水问题

人工智能  
经典问题求解算法——  
求解方法

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



## 求解方法种类

状态搜索空间法

启发式搜索求解

遗传算法求解

模拟退火求解

约束满足求解

## 典型问题案例

八数码

走迷宫

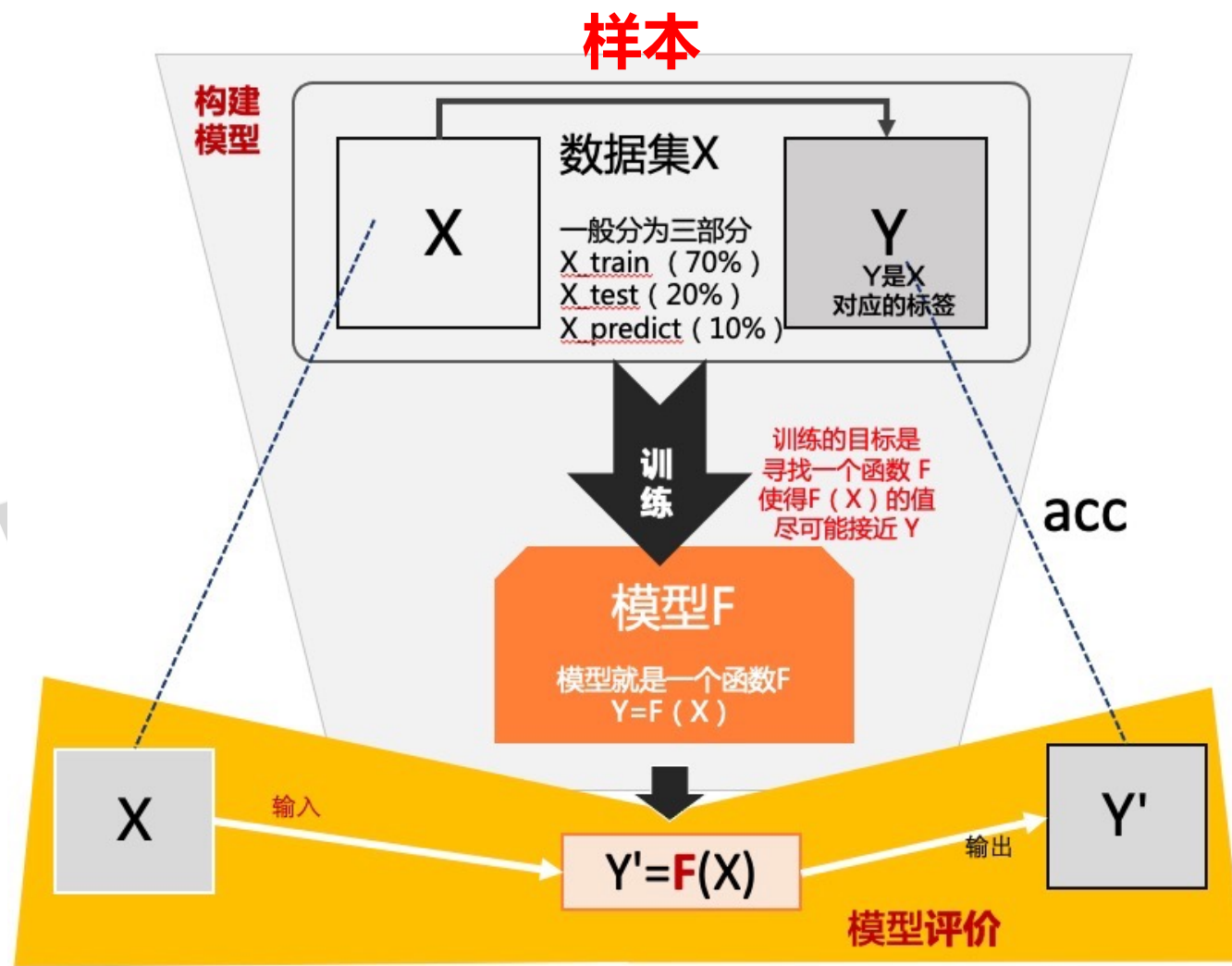
旅行推销员

拼图游戏

数独游戏

## □ 机器学习的定义：

- 已知给出一个数据集Dataset (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，Y'最大化接近Y。
- 我们把利用机器的算法寻找函数F的过程称为机器学习。
- 每一对X、Y又叫样本，总个数叫样本大小
- Y的所有可能取值叫样本空间，Dataset是样本空间的一个随机抽样



# 提纲

1

## 理解机器学习

2

## 回归学习

3

## 分类学习

4

## 模型训练与优化

5

## 模型评估与选择

# 1 理解机器学习与非机器学习



机器学习是人工智能的子集。传统问题求解过程是寻找算法解决问题的过程。机器学习一样，寻找**模型F**，即**算法**。

	机器学习	非机器学习
先验数据	必须	非必须
数学模型	人工设计	人工设定
模型参数	计算机自动习得	人工设计
准确率	难以100%	容易达到100%
推理	采用计算机	采用计算机

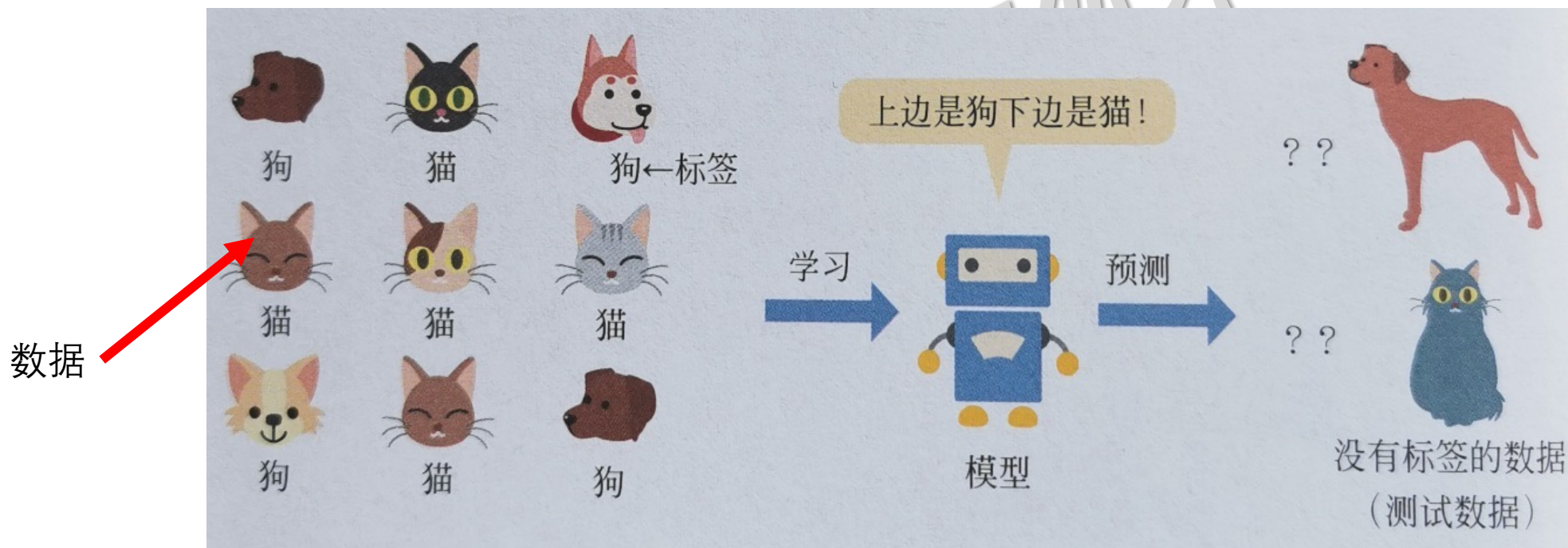


讨论：下列哪些属于机器学习

- 状态空间搜索
- 启发式搜索
- 最短路径算法
- 遗传算法
- 模拟退火
- 约束问题搜索
- 线性回归

## 监督学习

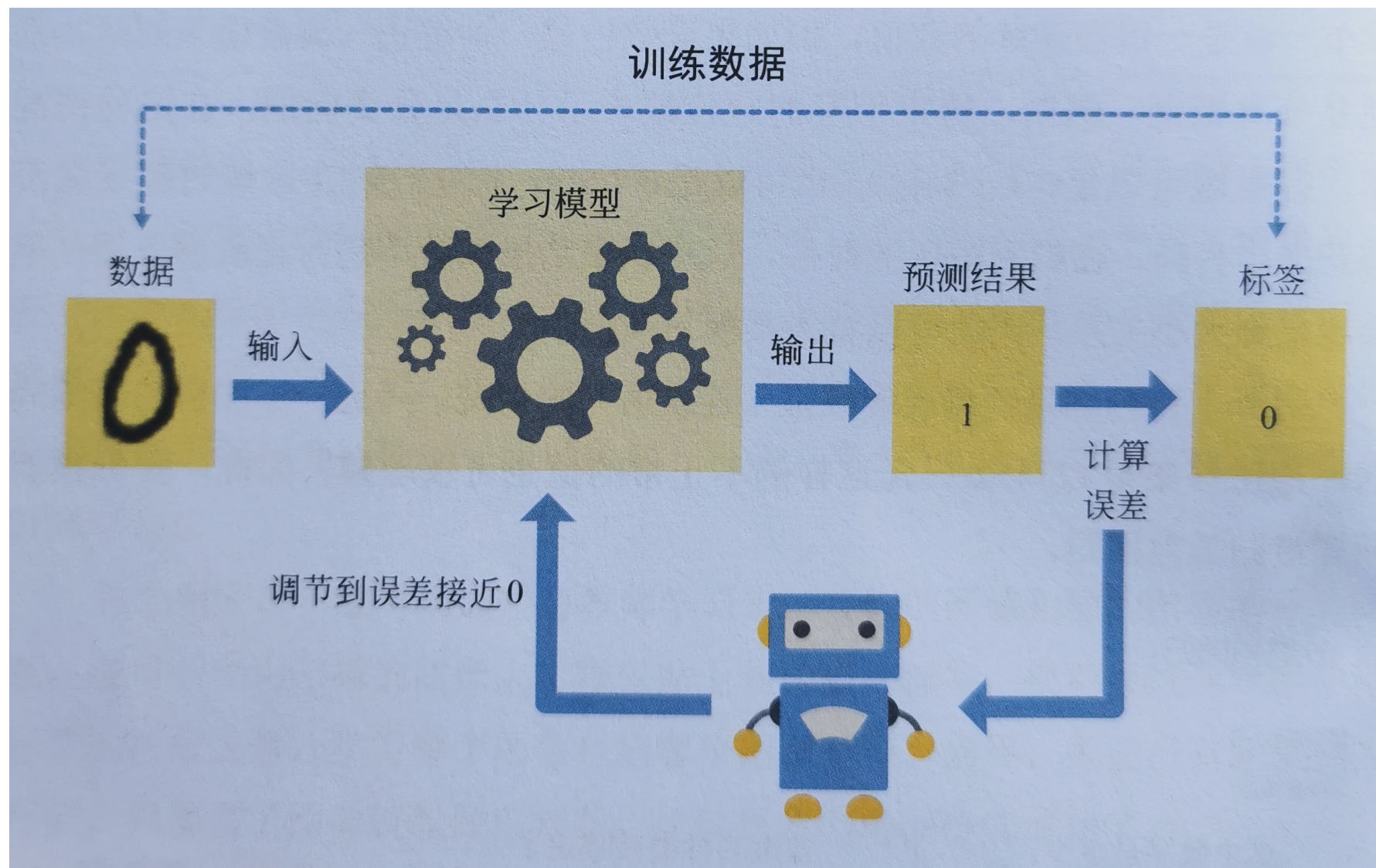
- 若所有的X都有Y对应，则是监督学习。
- 就是有老师带着你学习。
- 每个X，老师告诉你了Y，如何得到F。

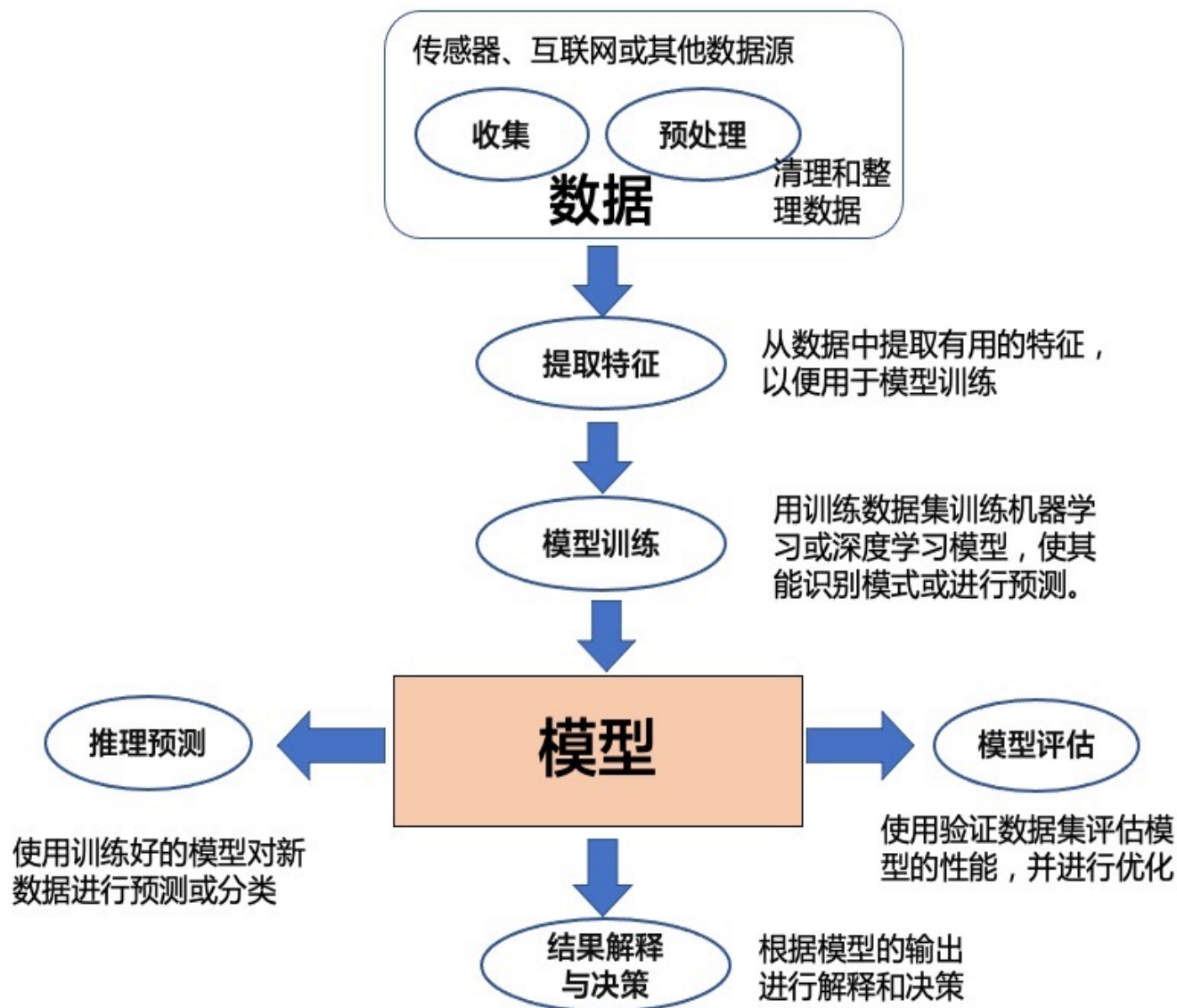




# 1 理解训练

训练的整个过程是不断降低误差的过程





## 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



## 有监督学习

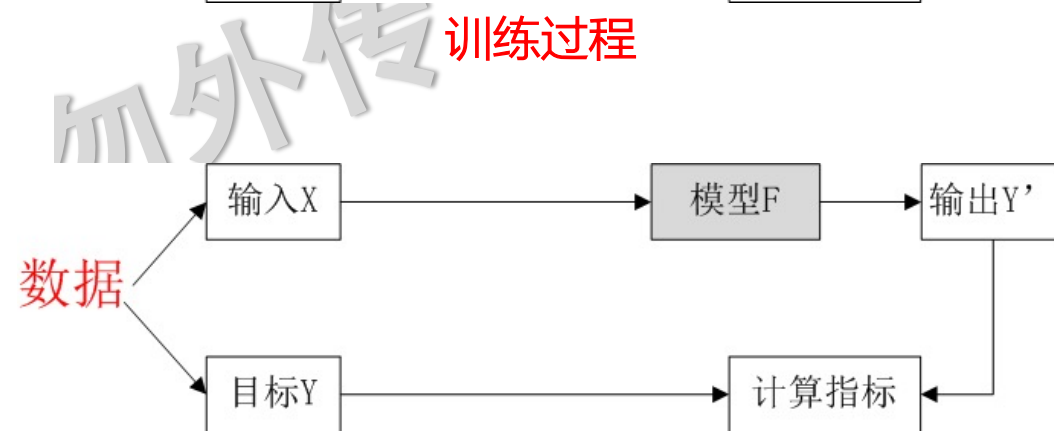
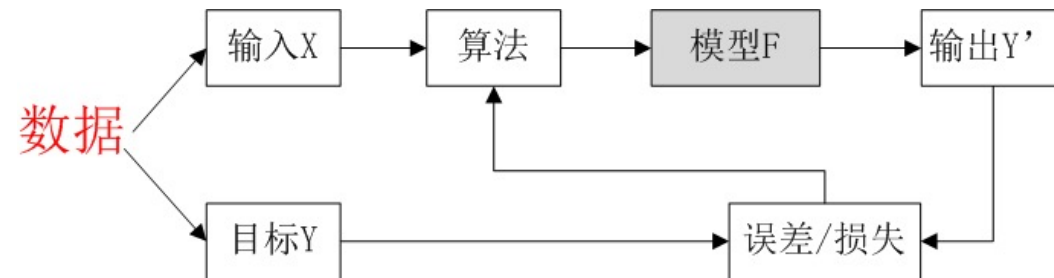
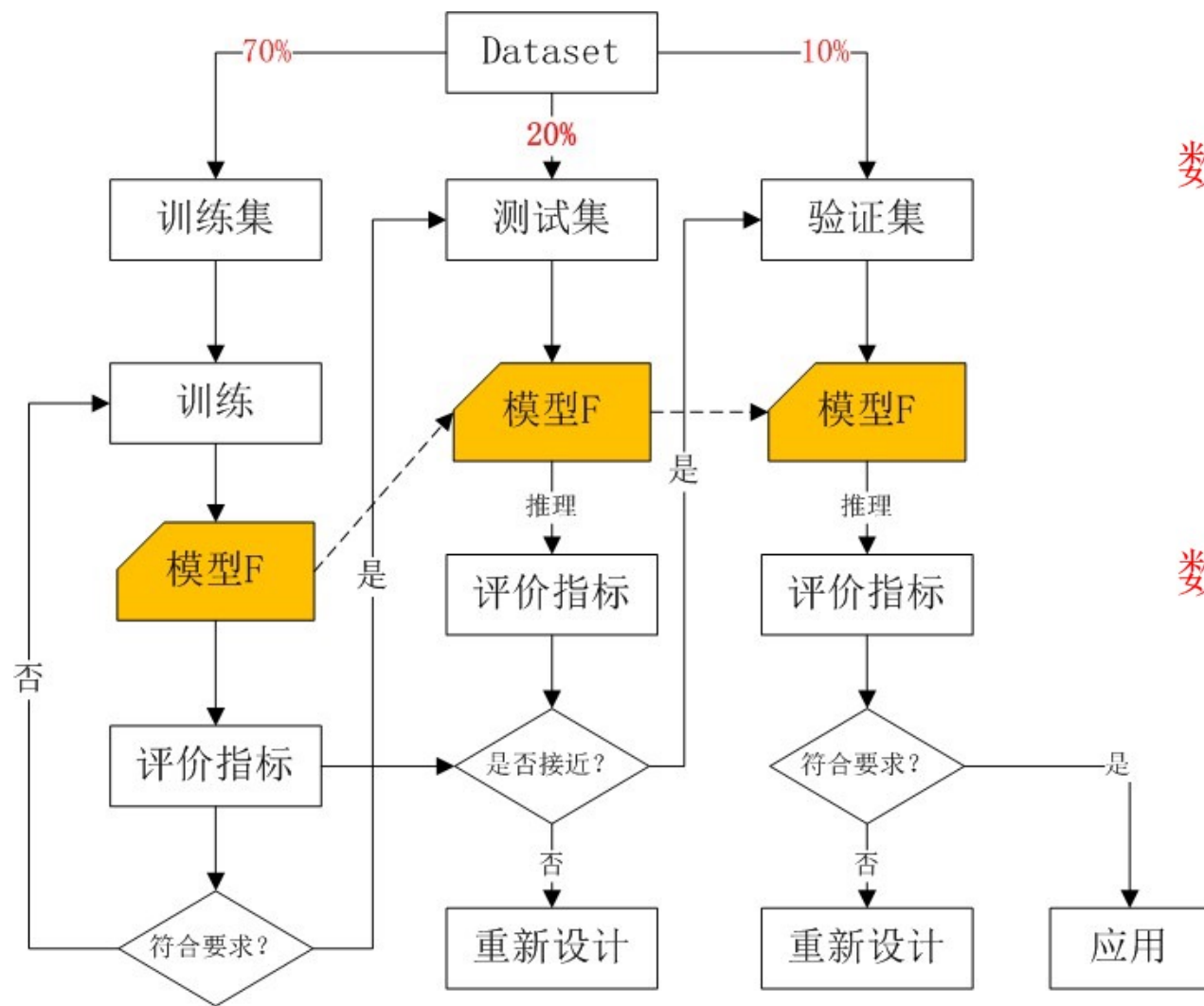
- 若所有的X都有Y对应，则是有监督学习。
- 就是有老师带着你学习。
- **每个X，老师告诉你了Y，如何得到F。**

## 无监督学习

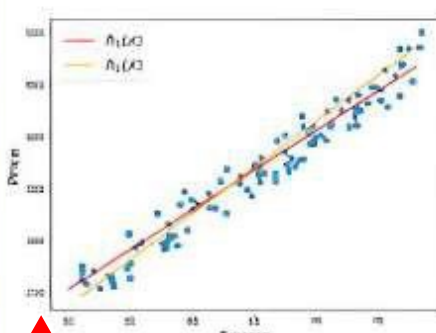


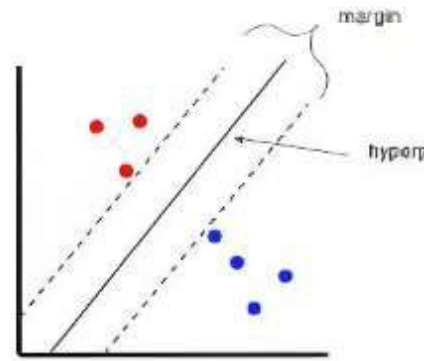
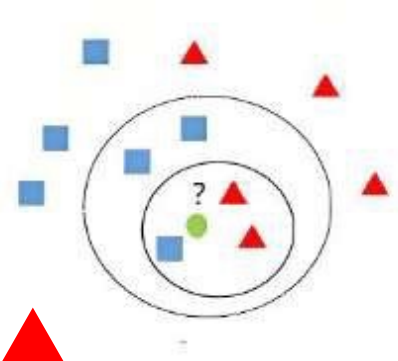
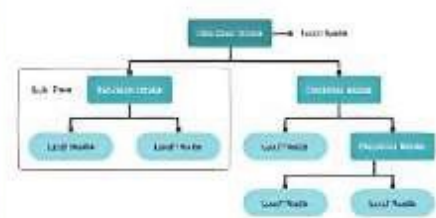
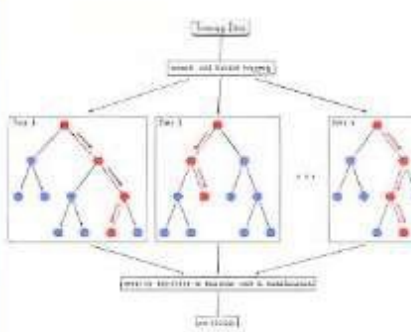
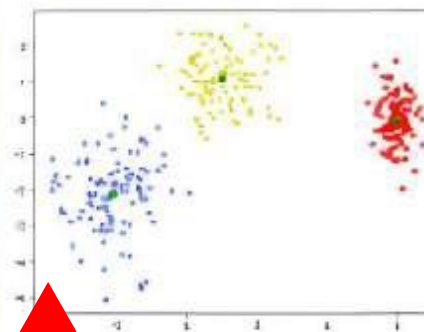
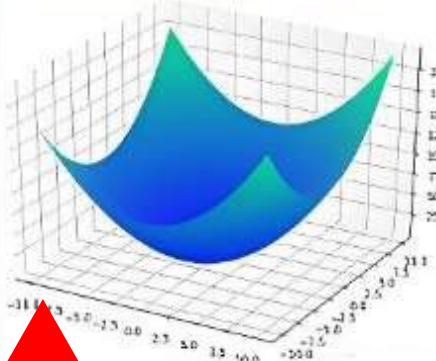
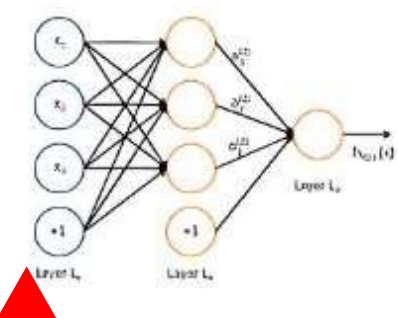
- 若没有Y，学习X内部知识，则称为无监督学习。
- 没有老师带着你，你自己学习。
- **X有了，老师没有告诉我Y，里面能有什么知识，如何得到F。**

## 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。
- **部分X、老师告诉你了Y，怎么得到F**



**五个要素**  
**数据、模型、训练、预测和评估**

<p><b>线性回归</b> Linear Regression</p> 	<p><b>逻辑回归</b> Logit Regression</p> 	<p><b>贝叶斯算法</b> Bayesian algorithm</p> 	<p><b>支持向量机</b> Support Vector machine</p> 	<p><b>K-近邻</b> k-Nearest Neighbor</p> 
<p><b>决策树</b> Decision tree</p> 	<p><b>随机森林</b> Random forest</p> 	<p><b>聚类算法</b> Clustering algorithm</p> 	<p><b>梯度下降</b> Gradient descent</p> 	<p><b>深度学习</b> Deep Learning</p> 



# 1 机器学习算法分类回归

## 分类

这个数字是多少？

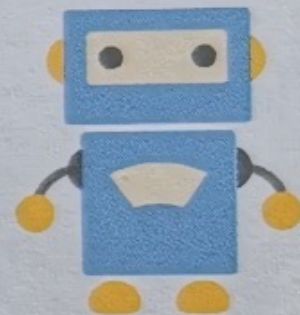
3

0? 3?

1? 2? 9?.....

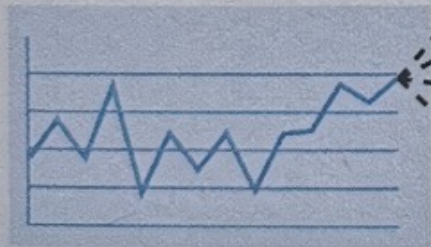
答案的种类被限定了  
(离散值)

3



## 回归

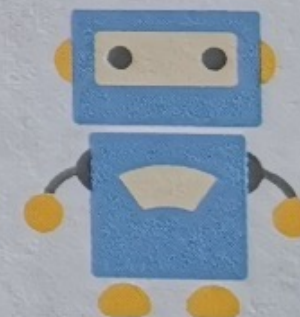
明天的股价是多少？



?  
?  
?  
?

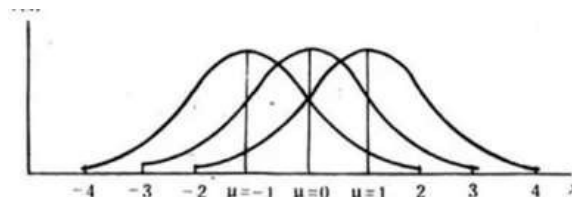
答案可以是任何值  
(连续值)

12345.6 日元

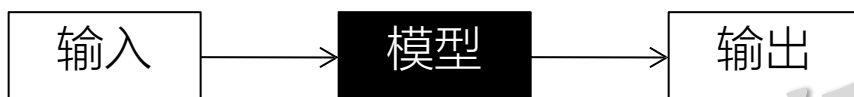




## 数据质量



## 可解释性



模型是个黑箱，模型越大，能力越强，可解释性越差。

方案：逆向工程的研究还没有突破性进展

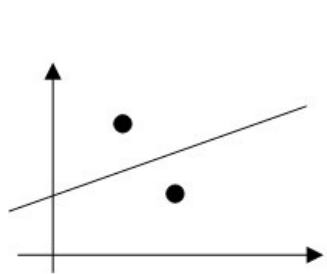
思考：可解释性真的必要吗？经验公式呢？

## 计算资源

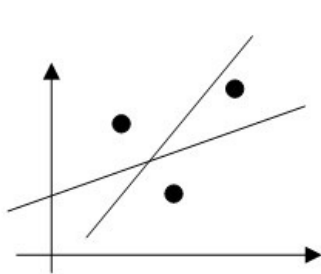
ChatGPT测算生成一条信息的成本在1.3美分左右，是目前传统搜索引擎的3到4倍

[GPT-4](#)的训练成本则达到了1亿美元

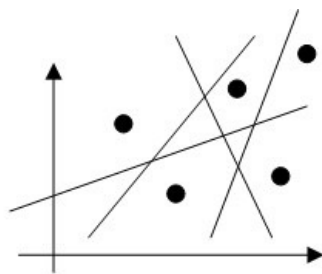
方案：设计更优的算法



1根线可分2个点

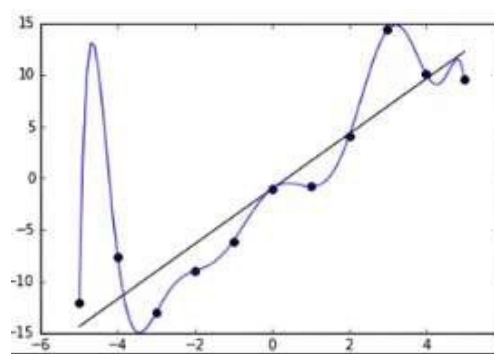


2根线可分3个点

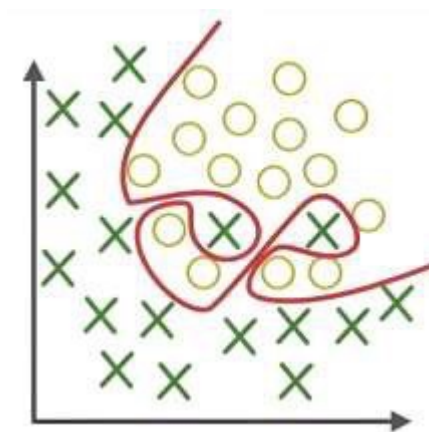


n根线可分n+1个点

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$$



非线性回归过拟合



**过拟合：训练集上表现很好，预测时表现很差的现象**

**思考：过拟合一定是不好的吗？**

**如何消除过拟合——提升泛化能力**

1. 交叉验证
2. dropout：动态随机切断神经元的连接
3. 正则化：L1范式（LASSO回归）和L2范式（岭回归），优化损失函数
4. 不过度训练（实时监控）
5. 特征选择（难度较大，往往事与愿违）
6. 集成学习，即多种不同质模型的组合投票
- 7. 核心：数据质量与预处理技术（正道）**

# 提纲

- 1 理解机器学习
- 2 回归学习
- 3 分类学习
- 4 模型训练与优化
- 5 模型评估与选择

## 2 回归

**定义：** 回归问题指的是预测连续数值输出的任务，目标是找出自变量 $X$ 和因变量 $Y$ 之间的数学函数关系

**线性回归：**  $Y$ 是关于 $X$ 的一次函数（方程）， $X$ 可以是一元，也可以是多元

**非线性回归：**  $Y$ 是关于 $X$ 的 $N$ 次函数（方程）， $N > 1$ ， $X$ 可以是一元，也可以是多元

**股价预测：**



线性回归



**定义：** 线性回归是统计学和机器学习中最基础且应用广泛的方法之一。它不仅简单易懂，而且为更复杂的回归方法奠定了基础。

线性回归的核心思想是假设输入变量（自变量）和输出变量（因变量）之间存在线性关系。数学上，这种关系可以表示为：

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \epsilon$$

其中 $Y$ 是因变量（我们想要预测的值）， $X_1, X_2, \dots, X_n$ 是自变量（已知的特征）， $\beta_0$ 是截距， $\beta_1, \beta_2, \dots, \beta_n$ 是各个自变量的系数， $\epsilon$ 是误差项，代表模型无法解释的随机变动。

简单线性回归是指只有一个自变量。例如，预测房价仅基于面积，

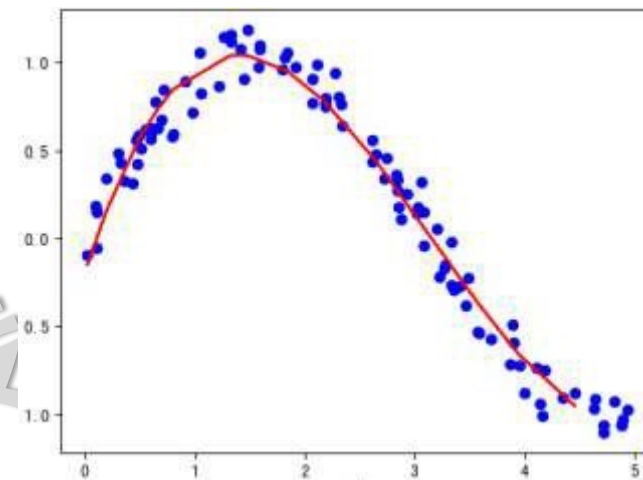
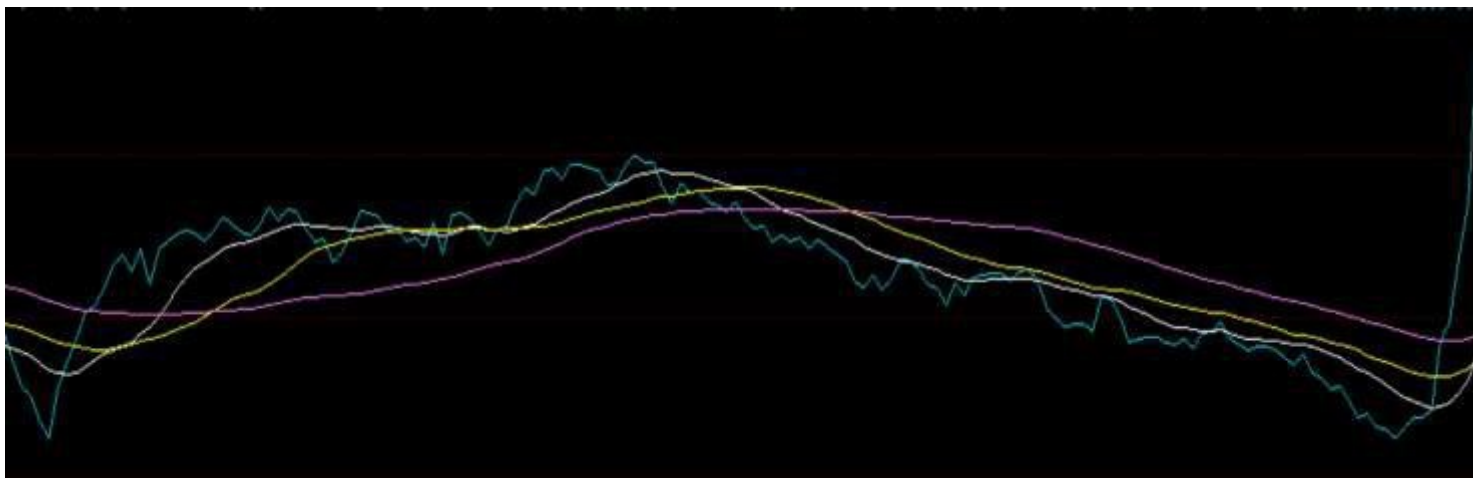
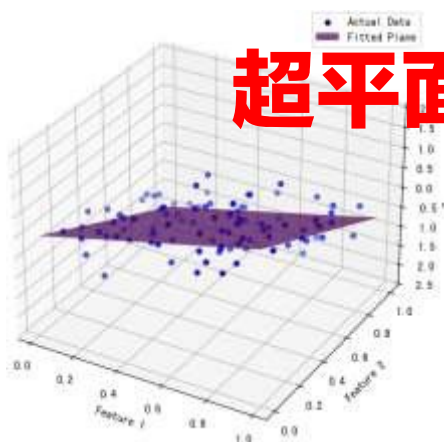
$$\text{房价} = \beta_0 + \beta_1 \times \text{面积} + \epsilon$$

多元线性回归是指有多个自变量。例如，预测房价基于多个因素，

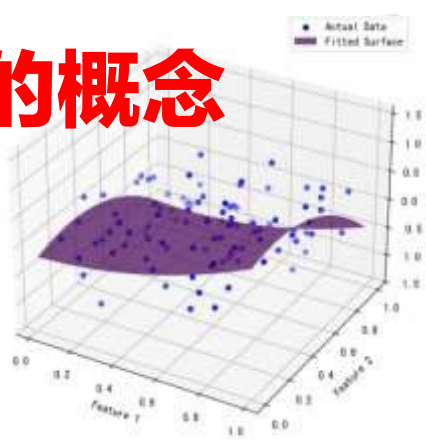
$$\text{房价} = \beta_0 + \beta_1 \times \text{面积} + \beta_2 \times \text{卧室数} + \beta_3 \times \text{距地铁站距离} + \epsilon$$

线性回归的目标是找到最佳的 $\beta$ 值，使预测值与实际值之间的差异最小。

非线性回归

**超平面的概念**

多元线性回归



多元非线性回归

用SKLearn实现回归算法-回归SKLearn.ipynb

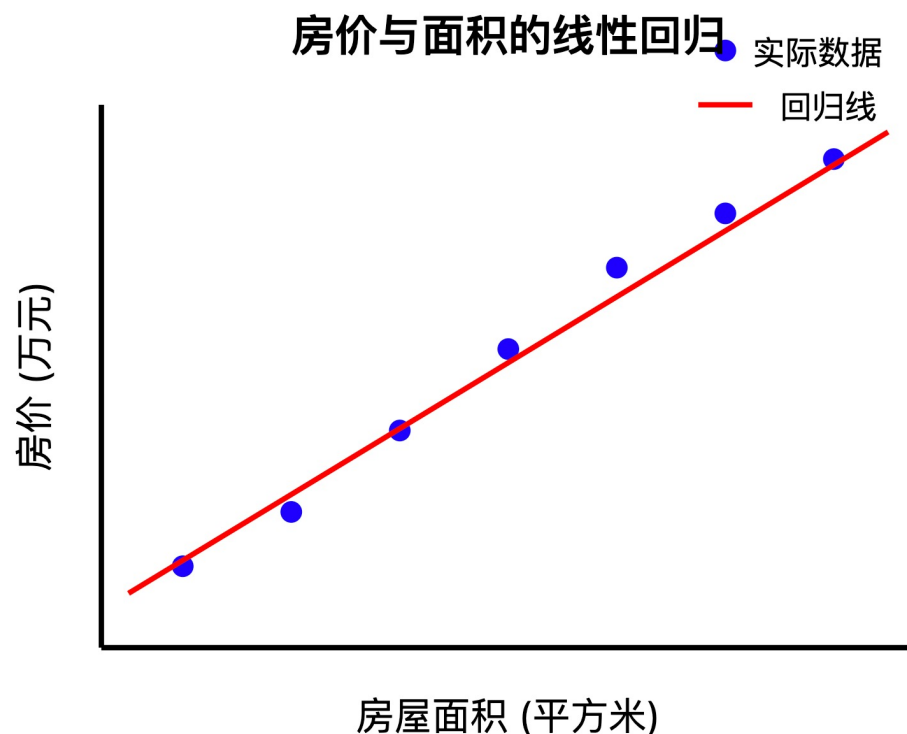
- 1、一元线性回归
- 2、非线性回归
- 3、多元线性回归
- 4、二元线性回归，并画出3D图
- 5、二元非线性回归，并画出3D图

## 2

## 案例1: 房价预测

让我们通过一个具体的房价预测例子来理解线性回归。

如下图所示，蓝点表示实际数据，每个点代表一个房屋样本的面积和价格。红线是回归线，表示模型预测的线性关系。这个图直观地展示了线性回归如何通过寻找最佳拟合线来描述两个变量（在此例中为房屋面积和价格）之间的关系。



## 1. 生成模拟数据

首先, 我们定义了一个generate\_house\_data函数来创建模拟的房价数据:

```
def generate_house_data(n_samples=1000):  
    area = np.random.uniform(50, 300, n_samples)  
    bedrooms = np.random.randint(1, 6, n_samples)  
  
    # 添加一些随机噪声  
    noise = np.random.normal(0, 20, n_samples)  
  
    # 生成价格 (单位: 万元)  
    price = -50 + 0.5 * area + 20 * bedrooms + noise  
  
    return pd.DataFrame({  
        '面积': area,  
        '卧室数': bedrooms,  
        '价格': price  
    })
```



## 2. 数据分割

接下来，我们将数据集分割为训练集和测试集：

```
X = df[['面积', '卧室数']]  
y = df['价格']  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

- X包含特征（面积和卧室数），y是目标变量（价格）。
- 使用train\_test\_split函数将数据分为80%的训练集和20%的测试集。
- random\_state=42确保结果可重复。

## 3. 训练模型

我们训练了两个模型：一个简单线性回归模型和一个多元线性回归模型。

简单线性回归：

```
simple_model = LinearRegression()  
simple_model.fit(X_train[['面积']], y_train)
```

这个模型只使用“面积”作为特征来预测房价。

多元线性回归:

```
multi_model = LinearRegression()
multi_model.fit(X_train, y_train)
```

这个模型同时使用"面积"和"卧室数"作为特征来预测房价。

#### 4. 模型预测

最后, 我们使用训练好的模型在测试集上进行预测:

```
y_pred_simple = simple_model.predict(X_test[['面积']])
y_pred_multi = multi_model.predict(X_test)
```

这些预测结果可以用于后续模型评估和比较。

# 提纲

- 1 理解机器学习
- 2 回归学习
- 3 分类学习
- 4 模型训练与优化
- 5 模型评估与选择

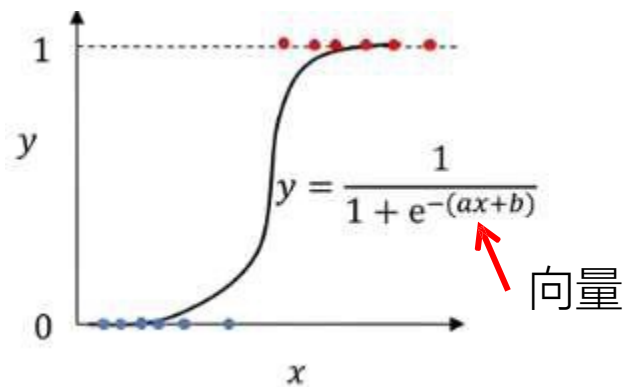
### 3 逻辑回归——二分类



**逻辑回归 (Logistic Regression)** 是一种**广义**的线性回归分析模型，也称为对数几率回归，它主要用于二分类问题，通过给定的自变量数据集来估计事件的发生概率。逻辑回归的因变量范围在0和1之间

函数变换的目的是去除噪声

设定一个阈值（如0.5），当因变量大于阈值时，为1  
否则为0，完成分类，



**补充概念：向量，转置，矩阵，矩阵相乘，降维**

$$\mathbf{a}^T = [a_1, a_2, a_3, \dots, a_n]$$

$$\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$$

$$\mathbf{ax} = a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n$$

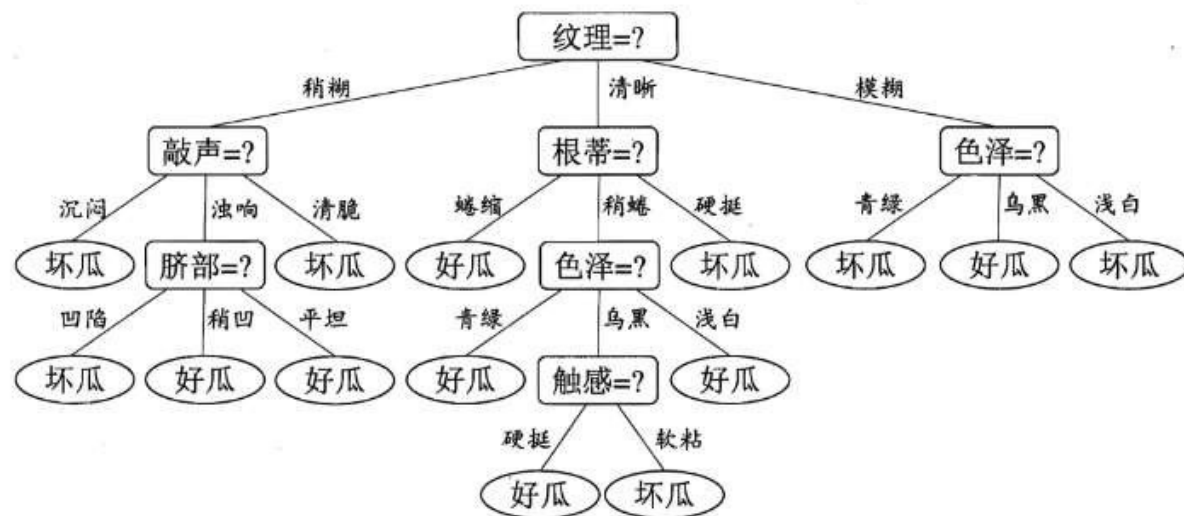
$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \quad B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

$$C = AB = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + a_{1,3}b_{3,1}, & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} + a_{1,3}b_{3,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} + a_{2,3}b_{3,1}, & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} + a_{2,3}b_{3,2} \end{bmatrix}$$

用SKLearn实现逻辑回归







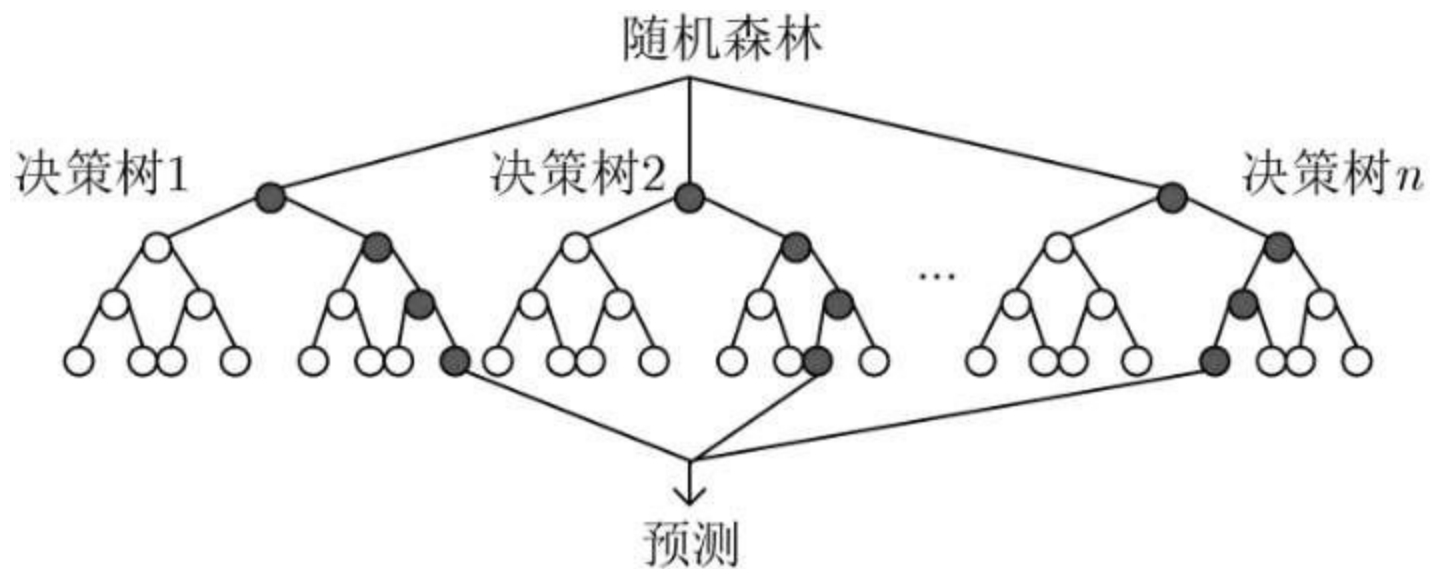
编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

### 原理

1. 计算每个特征分支下的子集正例和反例的概率
2. 通过概率计算子集的信息熵和特征的信息增益
3. 选择信息增益最大的特征作为根节点
4. 依次循环，只到划分完毕，构建了决策树
5. 搜索算法预测

-----纹理-----

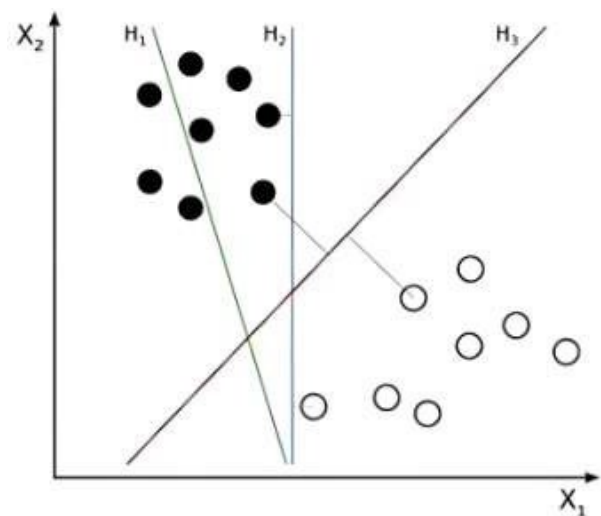
清晰 D1= ( 1,2,3,4,5,6,8,10,15 )	7:2	信息熵	→ 信息增益
稍糊 D2= ( 7,9,13,14,17 )	1:4	信息熵	
模糊 D3= ( 11,12,16 )	0:3	信息熵	



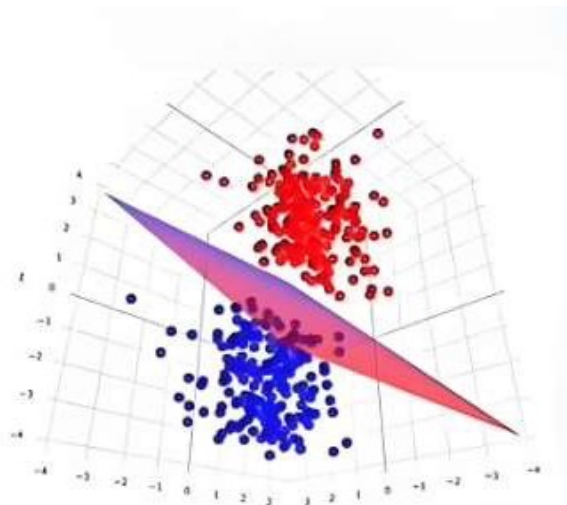
教学  
原理

1. 不同的决策树构建方式会得出不同的搜索结果
2.  $n$ 个决策树的结果进行投票

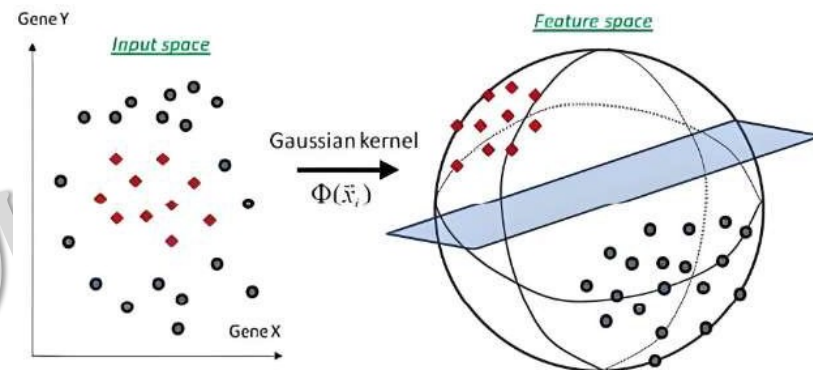
### 3 支持向量机SVM



二维平面



三维立体

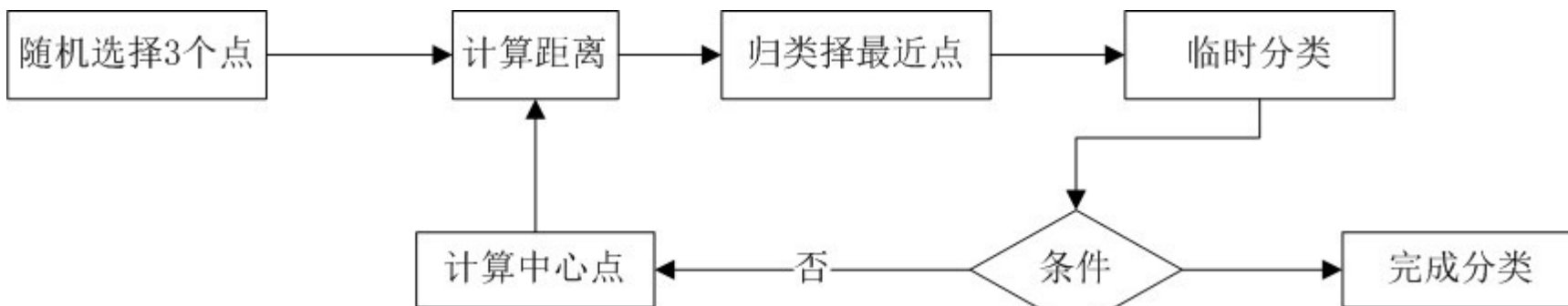
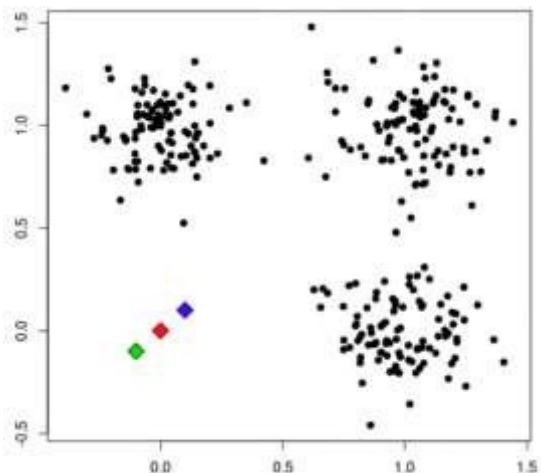


低维映射到高维

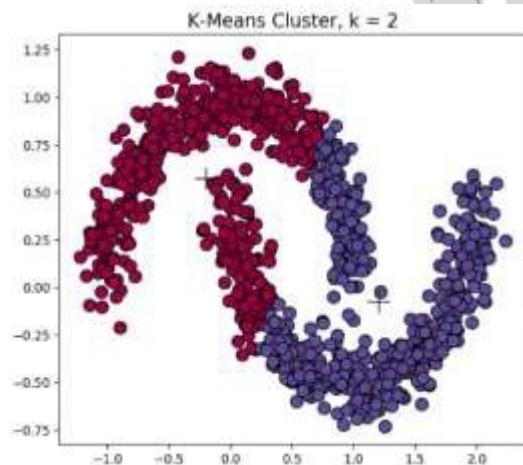
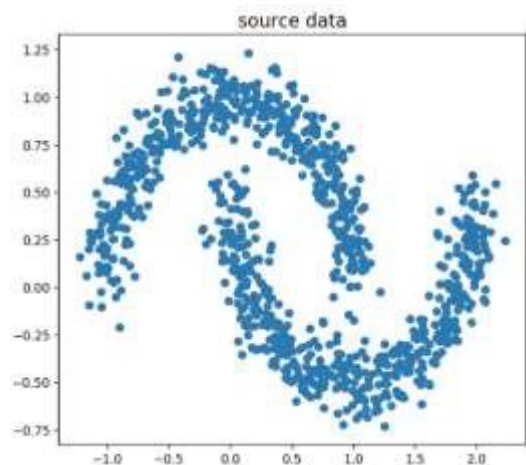
#### 原理

1. 找到一个超平面，使得不同分类的最近点距离这个平面均最远
2. 注意噪声的干扰

### 3 聚类、K-means、K-最近邻



K-means 算法 ( K=3 ) 无监督学习



#### KNN ( K-Nearest Neighbors ) -监督学习

通过计算待分类点与训练集中每个点之间的距离，选择距离最近的k个点作为邻居，然后根据这些邻居的类别进行投票，最终确定待分类点的类别。

#### 区别

- |                    |                    |
|--------------------|--------------------|
| 1. K的含义不同，K分类-K 邻居 | 3. 分类算法不同，KNN预测点归类 |
| 2. 学习分类不同          | 4. K-means KNN     |

**思考：KNN的学习表现在哪？**



## 朴素贝叶斯公式

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \rightarrow \frac{P(A)}{P(B)} = \frac{P(A|B)}{P(B|A)}$$

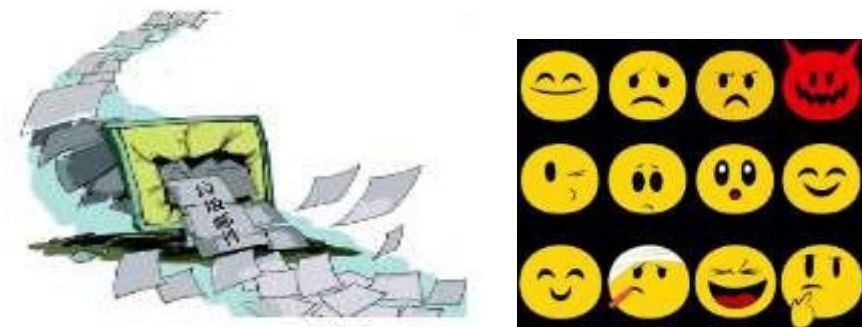
## 算法思想

预测：打开机器学（？）

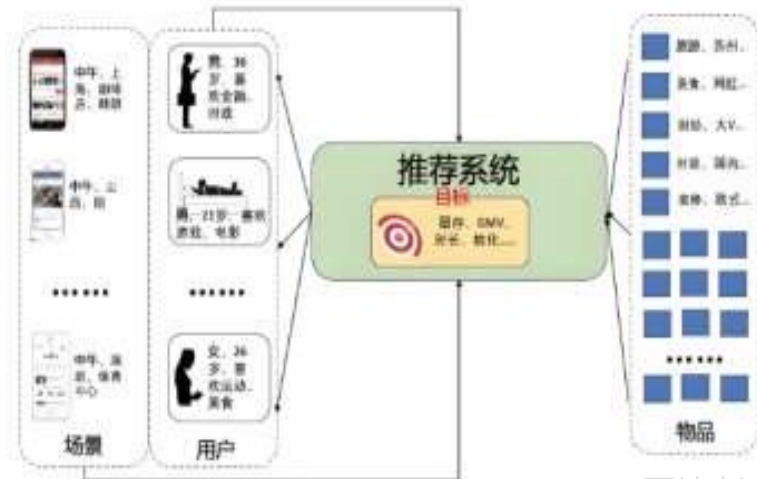
1. “学”之后出现什么字概率最大？ $B = \text{“学”}$
2. 通过语料统计所有字的字频，获得 $P(A)$ 、 $P(B)$
3. 通过语料计算所有AB同时出现的先验概率 $P(B|A)$
4. 计算所有A的 $P(A|B)$
5. 概率从大到小排序

## 几个概率名词

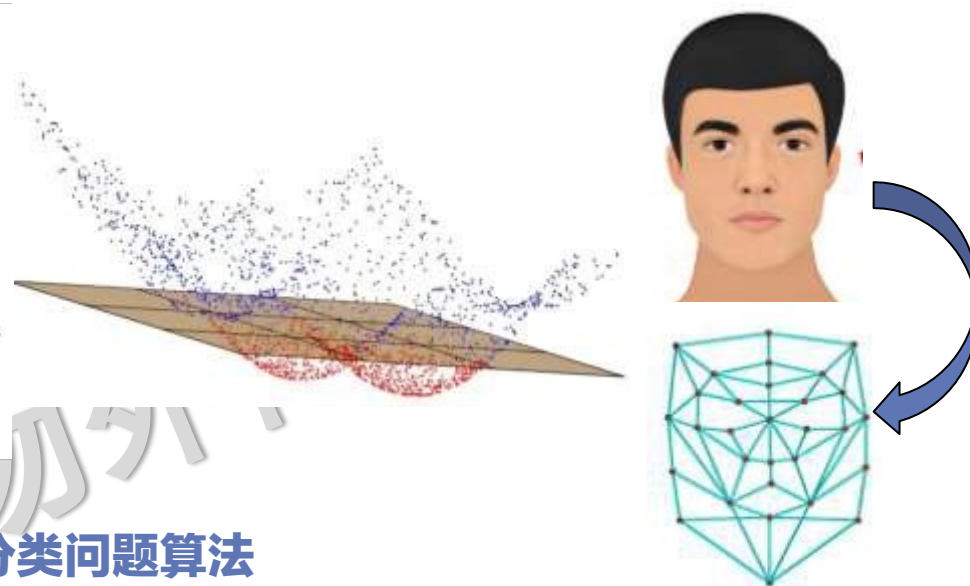
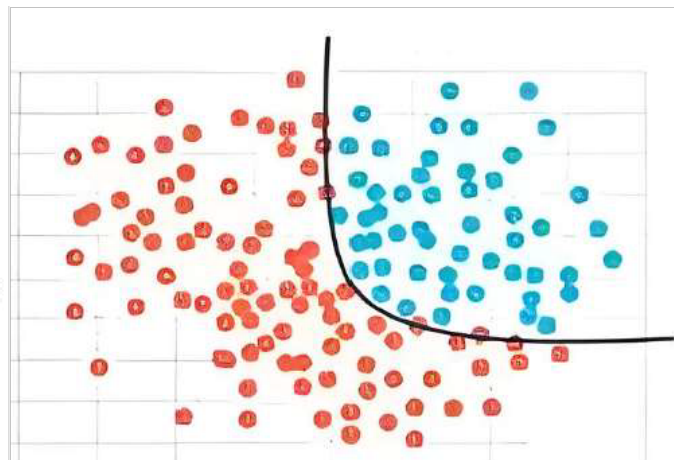
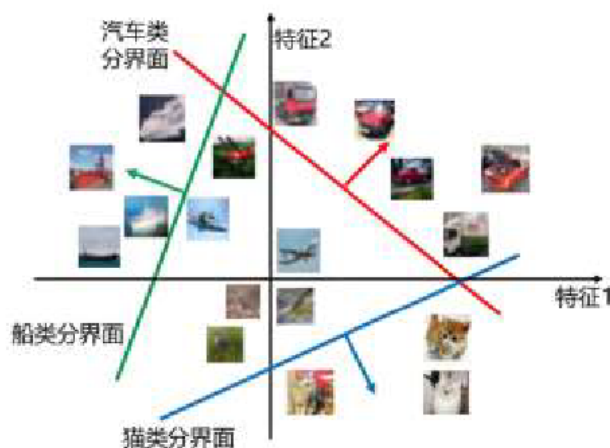
先验概率，条件概率，后验概率，独立事件（朴素）



应用领域







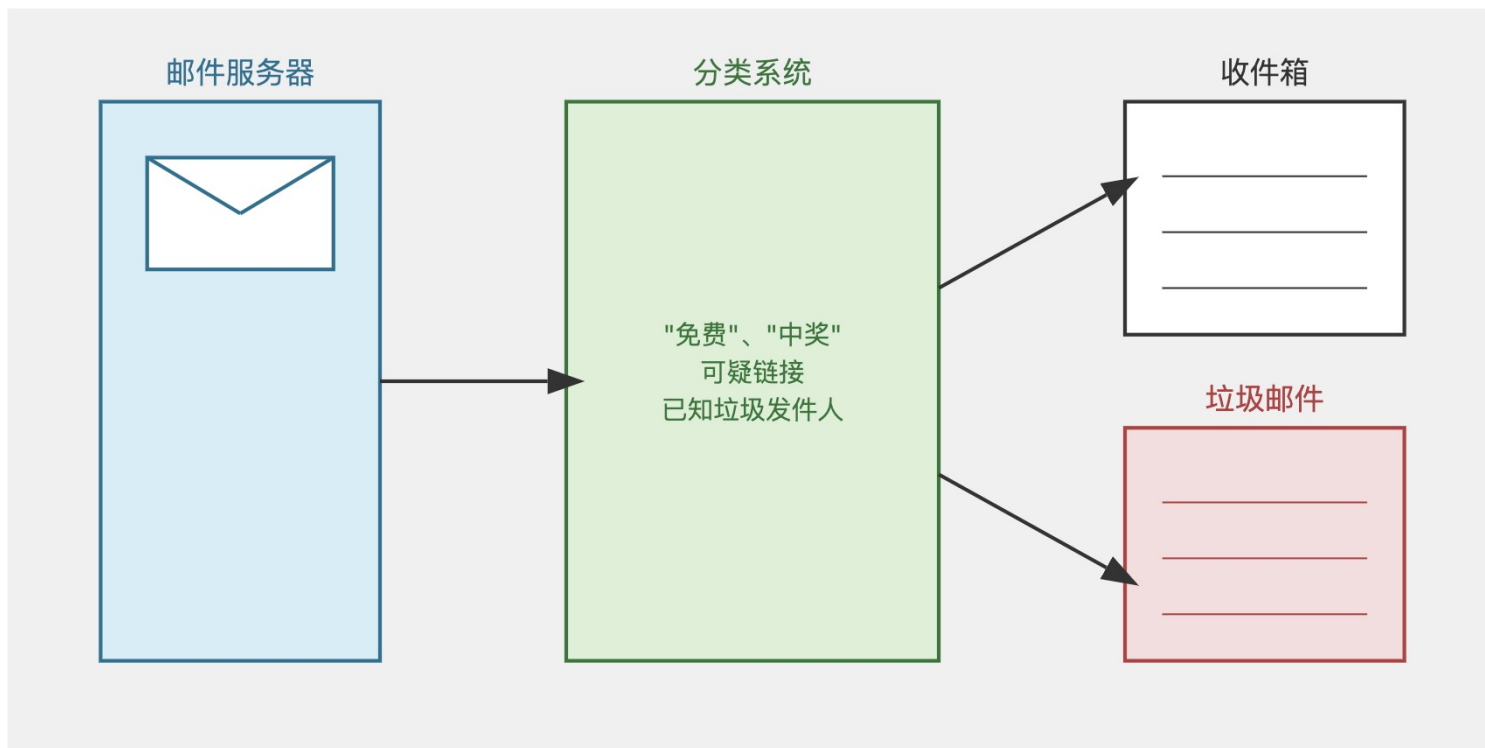
### 分类问题算法

1. 支持向量机 Support Vector Machine, SVM
2. 决策树 Decision Tree
3. 随机森林 Random Forest
4. 朴素贝叶斯 Naive Bayes
5. K最近邻 K-Nearest Neighbors, KNN
6. 深度学习 Deep Learning

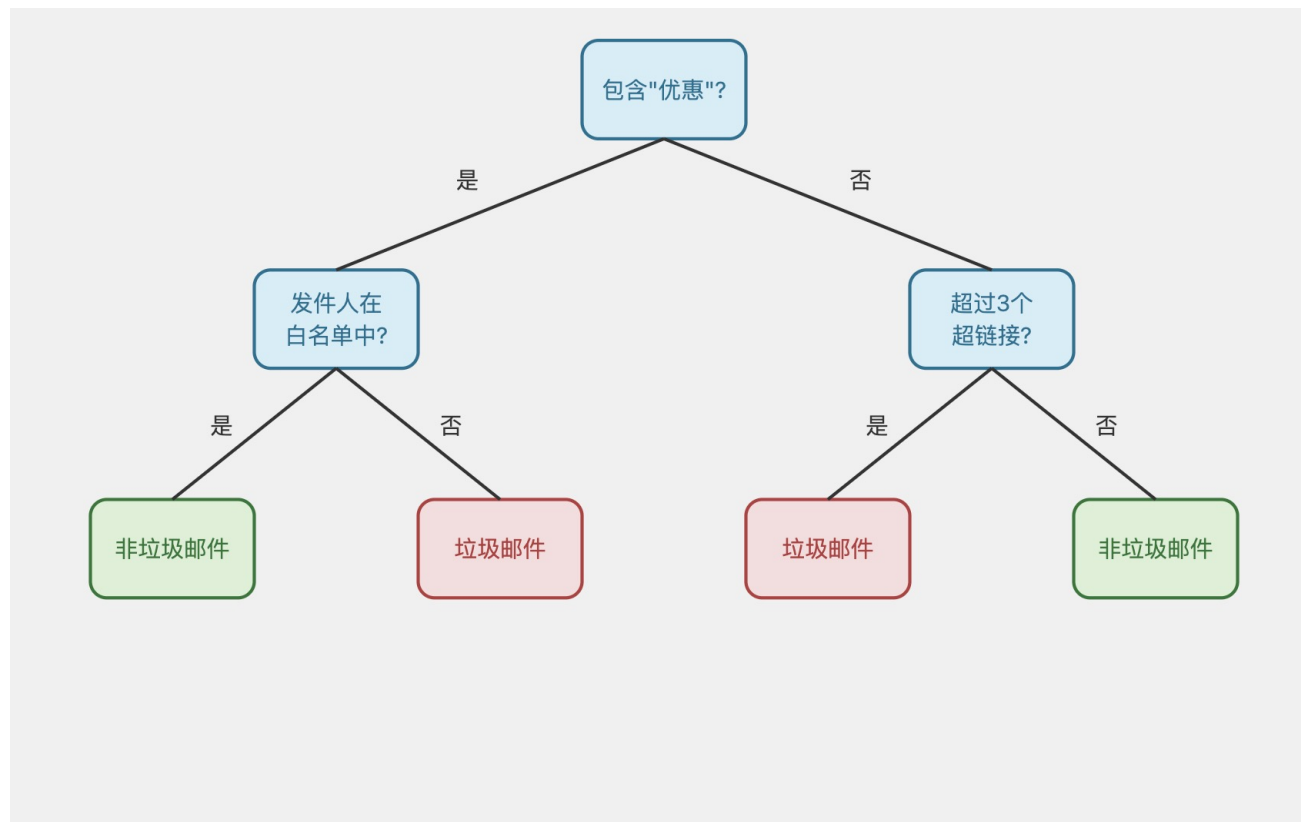


## 3 案例2：垃圾邮件识别

- 这是邮件从服务器到用户收件箱的整个过程
- 左侧代表邮件服务器，中间是基于机器学习的分类系统，右侧是最终的收件箱和垃圾邮件文件夹。
- 分类系统通过识别特定特征（如关键词"免费"、"中奖"，可疑链接，已知垃圾邮件发送者）来区分正常邮件和垃圾邮件。这个自动化过程不仅提高了效率，还减少了用户处理大量无用邮件的负担，展示了分类学习在解决实际问题中的应用。



## 3 案例2：垃圾邮件识别



- 上图展示了一个简单但有效的决策树，用于区分垃圾邮件和正常邮件。
- 决策过程从根节点开始，首先检查邮件是否包含"优惠"一词，然后根据结果进行进一步判断。
- 左侧分支检查发件人是否在白名单中，右侧分支则检查超链接数量。每个叶节点（绿色表示正常邮件，红色表示垃圾邮件）代表最终的分类结果。

## 3 案例2：垃圾邮件识别

以下代码实现了一个简单的垃圾邮件分类系统，使用决策树算法。让我们逐步分析代码的每个部分。

### 1. 数据准备

```
emails = [  
    {"content": "优惠大促销，限时抢购！", "is_spam": 1},  
    {"content": "您的账单已到期，请及时支付", "is_spam": 1},  
    # ... 其他邮件数据 ...  
]  
df = pd.DataFrame(emails)
```

这部分代码创建了一个模拟的邮件数据集。每个邮件都有两个属性：内容（content）和是否为垃圾邮件的标签（is\_spam）。使用pandas库将这些数据转换为DataFrame，便于后续处理。

## 3 案例2：垃圾邮件识别

### 2. 特征提取

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['content'])
y = df['is_spam']
```

这一步使用了CountVectorizer来将文本内容转换为数值特征。它的工作原理是：

- 创建一个词汇表，包含所有邮件中出现的唯一词语。
- 对每封邮件，计算词汇表中每个词的出现次数，形成一个向量。
- 这样，每封邮件就被转换成了一个数值向量，这个向量可以被机器学习算法处理。

### 3. 划分训练集和测试集

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

使用 train\_test\_split 函数将数据集分为训练集（80%）和测试集（20%）。这样做的目的是为了后续评估模型在未见过的数据上的表现。



## 3 案例2：垃圾邮件识别

### 4. 训练决策树模型

```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

这里使用了决策树分类器。决策树的工作原理是从根节点开始，根据特征的值进行分枝。在每个分支点，选择最能区分不同类别的特征。重复这个过程，直到到达叶节点，叶节点代表最终的分类结果。

### 5. 预测和评估

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"准确率: {accuracy:.2f}")
print("\n 分类报告:")
print(classification_report(y_test, y_pred))
```

最后，使用训练好的模型对测试集进行预测，并计算准确率。`classification_report` 函数提供了更详细的评估指标，包括精确率、召回率和 F1 分数。

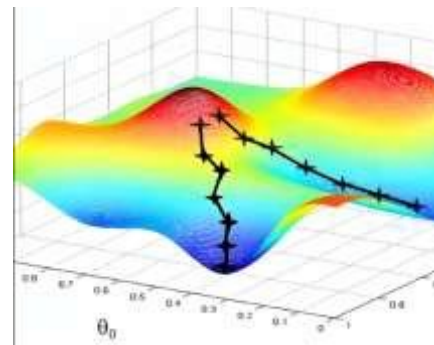
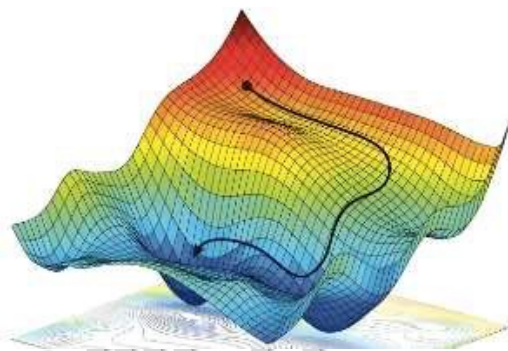
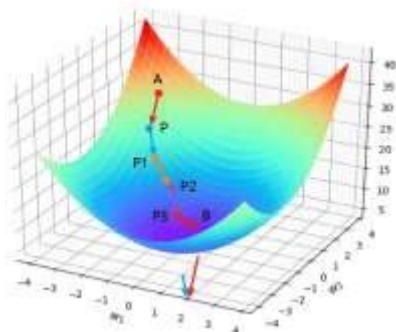
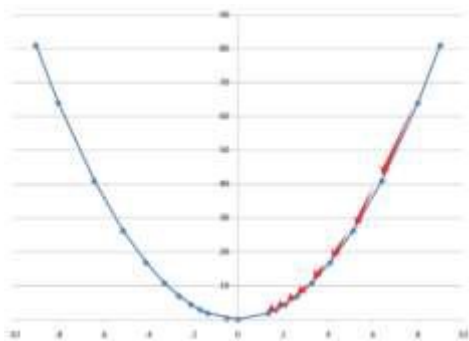
# 提纲

- 1 理解机器学习
- 2 回归学习
- 3 分类学习
- 4 模型训练与优化
- 5 模型评估与选择

## 4

## 梯度下降

定义：以最快的速度找到函数局部最小值的优化算法



梯度 $\nabla$ ：函数在某点变化最快的方向一元

方程：某点的斜率（导数）；

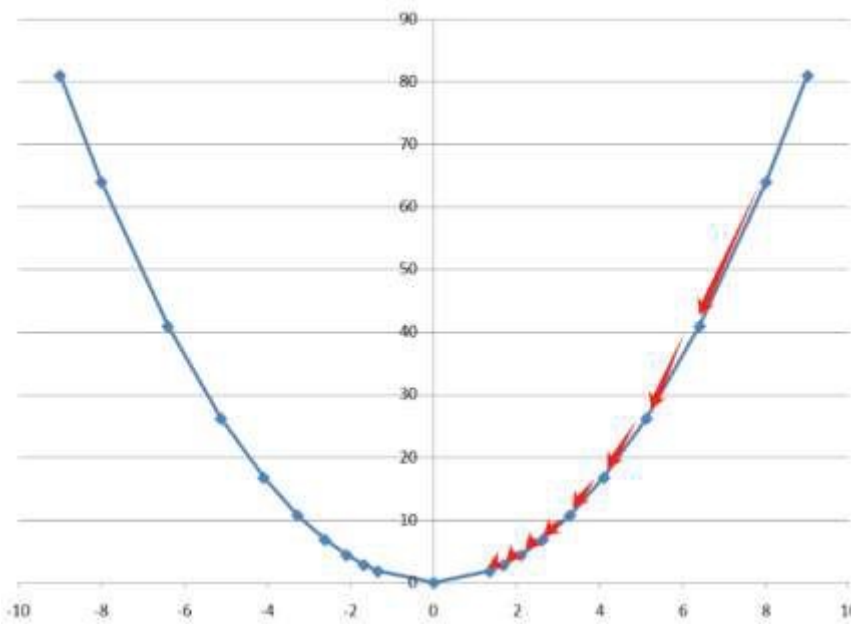
多元方程：函数所有变量在某处偏导数的向量。

$$\mathbf{grad} f(x, y, z) = \nabla f(x, y, z) = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right]$$

最小值寻找过程：（1）方向的选择-梯度 （2）前进的步长-学习率 （3）何时终止 - 梯度 $\approx 0$

如何应用：设计损失函数，沿着损失函数梯度的反方向更新网络参数——误差反向传播（BP）算法

## 4 梯度下降案例



$$f(x) = x^2$$

起点P(8,64), 学习率=0.1,

$x_1$	8	6.4	5.12	4.096	3.277	2.622	2.098	1.679	1.343
$\nabla f$	16	12.8	10.24	8.192	6.554	5.244	4.196	3.358	.....
$x_1 - \eta \nabla f$	6.4	5.12	4.096	3.277	2.622	2.098	1.679	1.343	.....

练一练：学习率=0.2时，上述曲线梯度下降算法的过程，并画图（手工计算）

练一练：用 python 实现一个梯度下降算法的函数，参数为终止条件、学习率、起点

例8-3: 对曲面  $f(x, y) = 3x^2 + 2y^2$  进行梯度下降算法求局部低点, 起点P(3,5), 学习率  $\eta = 0.3$ , 终止条件  $\|\nabla\| < 0.002$ 。

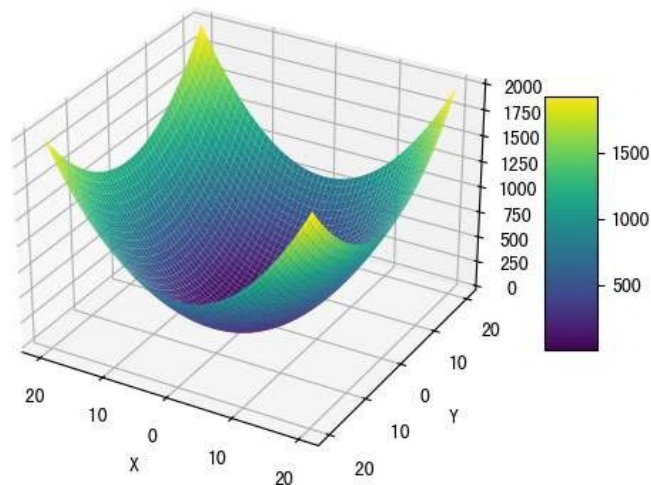
解:  $\frac{\partial f}{\partial x} = 6x$      $\frac{\partial f}{\partial y} = 4y$

轮次	x	y	$\frac{\partial f}{\partial x}$	$\frac{\partial f}{\partial y}$	$x - \eta \frac{\partial f}{\partial x}$	$y - \eta \frac{\partial f}{\partial y}$	$\ \nabla\ $
1	3	5	18	20	-2.4	-1	724
2	-2.4	-1	-14.4	-4	1.92	0.2	223.4
3	1.92	0.2	11.52	0.8	-1.54	-0.04	133.4
4	-1.54	-0.04	-9.22	-0.16	1.23	$8 \times 10^{-3}$	85.0
5	1.23	$8 \times 10^{-3}$	7.37	0.03	-0.98	$-1.6 \times 10^{-3}$	54.4
6	-0.98	$-1.6 \times 10^{-3}$	-5.91	$-6.4 \times 10^{-3}$	0.79	$3.2 \times 10^{-4}$	34.8
7	0.79	$3.2 \times 10^{-4}$	4.72	$-1.28 \times 10^{-3}$	-0.63	$-6.4 \times 10^{-5}$	22.3

依次计算, 直到符合条件

### 梯度下降法问题与限制

- 1、固定学习率-过犹不及或遥遥无期
- 2、局部最优
- 3、每次计算全部样本误差
- 4、损失函数必须连续可导





## 4 优化器-梯度下降的优化

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

一次计算全部样本的误差之和，一次只计算一个（批）

#### 优点

1. 高效
2. 可并行计算
3. 可适应新数据变化（预训练思想萌芽）
4. 有机会全局最优

#### 局限

1. 不稳定
2. 没有解决学习率选择问题
3. 随机最优解
4. 模型不可控

**改进**

- 小批量梯度下降法（Mini-batch SGD）
- 动量梯度下降法（Momentum SGD）

#### Python 实现

```
optimizer = optim.SGD(model.parameters(), lr=0.01)
optimizer = optim.SGD(model.parameters, momentum=0.9)
```

### 自适应梯度算法 Adaptive Gradient, AdaGrad

根据权重的梯度自适应调整学习率

#### 优点

1. 自动化调整学习率
2. 自适应，所有权重“步调一致”

#### 局限

1. 梯度消失（训练次数作为分母）
2. 训练速度慢

**改进**

- 自适应平方根梯度法（RMSProp）
- 自适应矩估计法（Adam）目前最常用

#### Python 实现

```
optimizer = optim.Adagrad(model.parameters(), lr=0.01, weight_decay=1e-4, eps=1e-10)
optimizer = optim.Adam(model.parameters(), lr=0.01, beta_s=(0.9, 0.999), eps=1e-8, weight_decay=0)
```

# 提纲

- 1 理解机器学习
- 2 回归学习
- 3 分类学习
- 4 模型训练与优化
- 5 模型评估与选择

5

	预测正	预测负
实际正	TP：真正例	FN：假反例
实际负	FP：假正例	TN：真反例

$$\text{准确率} = \frac{\text{预测正确的样本数}}{\text{样本总数}} \times 100\%$$

**准确率**并不适用于所有情况，特别是在样本类别不平衡时。

$$\text{精确率} = \frac{TP}{TP + FP} \times 100\%$$

**精确率**适用于重视准确预测正例的情况，例如疾病预测等。

**基准测试:** 与公认的大型数据集进行对比验证。  
公认的数据集一般经过数据清洗，具有较好的分布

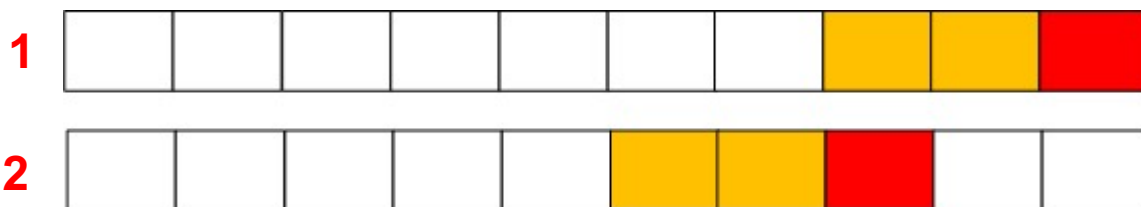
$$\text{召回率} = \frac{TP}{TP + FN} \times 100\%$$

**召回率**适用于重视将所有正例样本预测出来的情况，如目标检测。

$$F1 = 2 \times \frac{\text{精确率} \times \text{召回率}}{\text{精确率} + \text{召回率}}$$

**F1**用于衡量分类模型的整体性能，值越高，表示模型在准确率和召回率之间取得了平衡

**交叉验证:** 这是一种常用的模型评估方法，通过将数据集划分为多个子集，将模型训练和测试分别在不同的子集上进行，从而减少过拟合和提高模型性能。



## 5

# 过拟合与欠拟合

## 过拟合

模型在训练数据上表现良好，但在测试数据或新数据上表现较差。

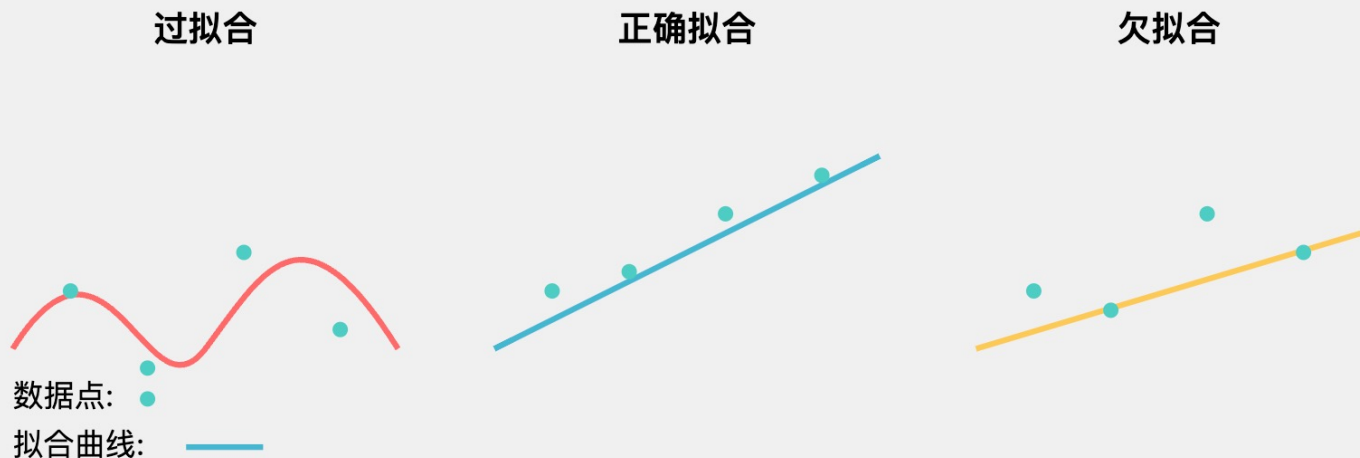
- **正则化**：通过引入惩罚项来限制模型的复杂度（L1正则化、L2正则化）。
- **交叉验证**：使用交叉验证方法评估模型的泛化能力。
- **减少模型复杂度**：选择简单的模型，避免过于复杂的非线性结构。

## 欠拟合

模型过于简单，无法捕捉数据中的关键模式。

- **增加模型复杂度**：使用更复杂的模型或添加更多的特征。
- **增加特征**：添加更多有意义的输入特征，帮助模型更好地捕捉模式。

### 过拟合 vs 正确拟合 vs 欠拟合



过拟合: 复杂模型, 完美拟合训练数据但泛化性差

正确拟合: 平衡复杂度, 良好拟合数据并具有泛化性

欠拟合: 过于简单, 无法捕捉数据的复杂性

## 交叉验证 (Cross-Validation)

将数据集分成多个部分，依次将每个部分作为验证集，其余部分作为训练集，重复多次并取平均结果，确保模型的评估更加稳健。

- **常用方法**：k折交叉验证 (k-fold cross-validation) 是最常见的方法，将数据分成k份，进行k次训练和验证，取平均性能作为评估结果。
- **优势**：交叉验证可以有效防止模型过拟合，并帮助选择最佳的模型参数。

## 模型选择

- **超参数调整**：通过网格搜索 (Grid Search) 或随机搜索 (Random Search) 找到最佳超参数组合，如学习率、正则化参数等。
- **模型性能与计算效率的平衡**：
  - 在某些应用中，性能和准确性是首要考量，如金融或医疗领域。
  - 在实时应用中，模型的计算效率也是重要考虑因素，性能与效率的平衡需要通过测试来实现。



