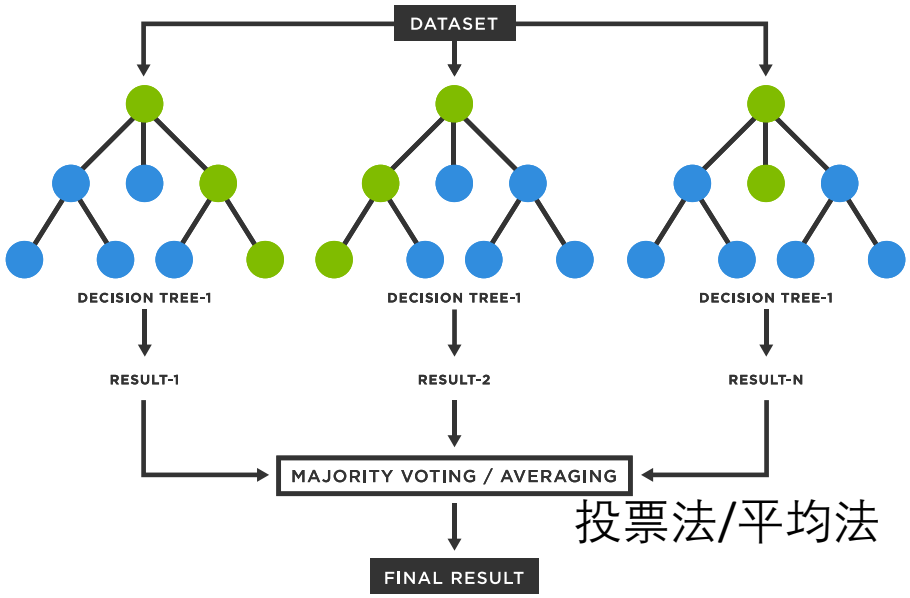
随机森林

随机森林

- "随机森林（Random Forests, RF）是由多棵决策树组成的集成学习方法，每棵树独立地对数据进行分类或回归预测，最终通过投票或平均的方式得到最终预测结果。
- 随机森林计算开销较小，并且在许多任务中表现出卓越的性能，因此被誉为“集成学习的代表方法”。
- 森林?



随机森林的两大优势（相对于单个决策树）

- **降低方差**：集成决策树通过对多个决策树的预测结果进行组合（如投票或平均），**平滑**了每棵树的预测波动，减少了整体模型对训练数据细微变化的敏感性，从而降低了方差。

- **减少过拟合**：单棵决策树容易过拟合，因为它倾向于学习训练集中的每个细节。集成决策树中，每棵树的训练数据和模型结构都有差异，从而避免了单棵树的过拟合问题，最终通过集成多棵树的预测来减少整体模型的过拟合。

总结：每个决策树的错误各有各的不同，通过集成的“平均效应”，将错误“平滑”了，所以能够降低方差，减少过拟合。

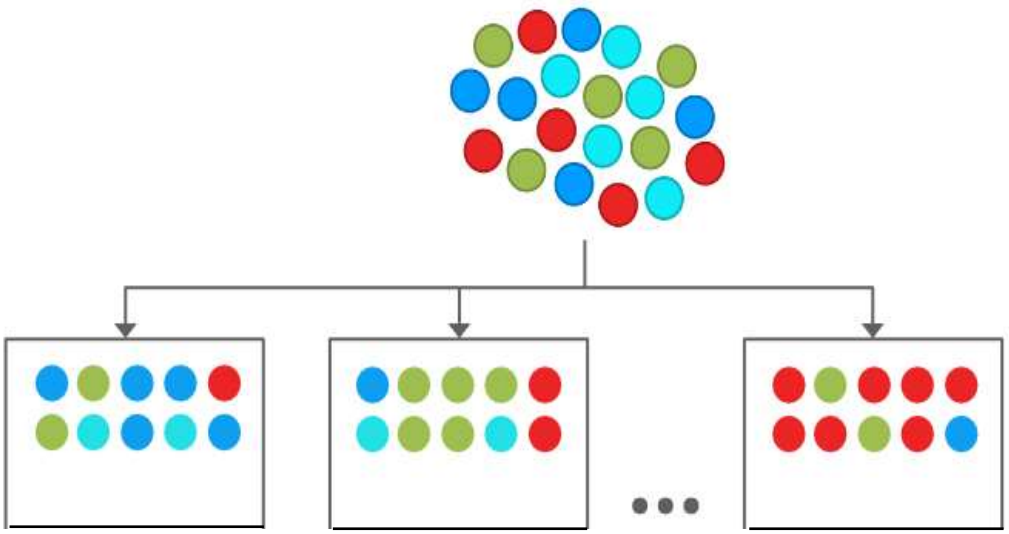
随机森林的“随机”

- 在随机森林算法中，需要注意单个决策树之间的相关性问题。如果多个决策树在同一数据集上训练，并且每棵树都选择相同的最佳分裂点，那么它们可能会变得高度相关，甚至完全相同。
- 为了提高集成模型的泛化能力，决策树之间应尽可能独立。随机森林通过以下方式引入随机性，从而减少树之间的相关性，确保它们能够提供多样化的信息：

- **数据**的随机性选取
- **节点特征**的随机选取

数据的随机选取

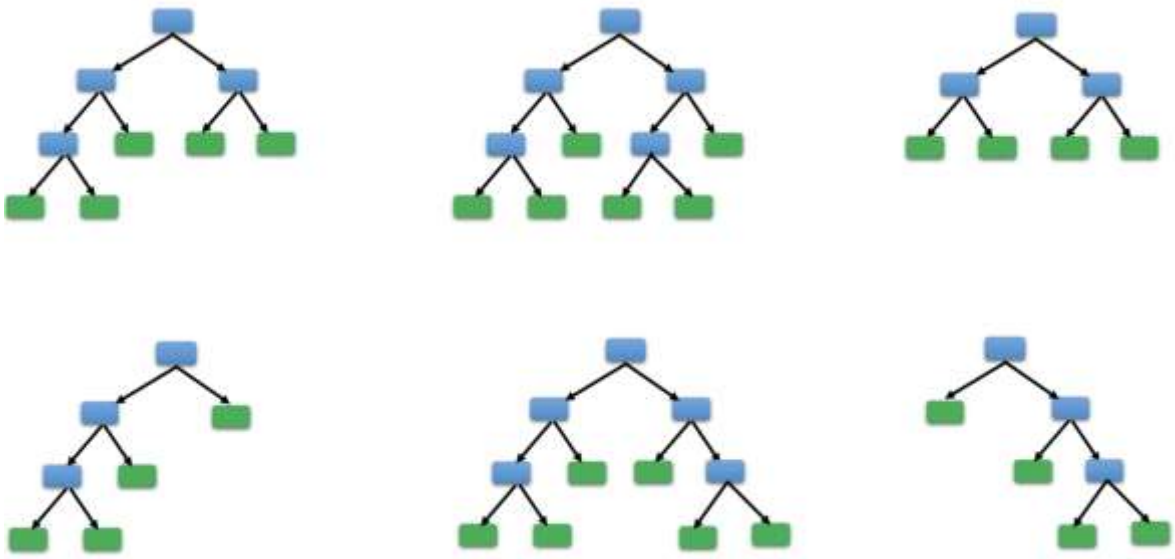
- **Bootstrap采样（Boostrap sampling）**主要用于从原始数据中通过重复抽样（带放回）生成多个新的样本集合，以进行估计、假设检验或评估模型的稳健性（robustness）。
- 随机森林使用 **Bootstrap采样** 来生成多个不同的样本子集：从包含N个样本的原始数据集中，随机抽取一个样本，并将其放回数据集，这个过程重复 **N次**。最终得到的样本子集大小为N，但由于有放回的抽样，某些样本可能会被重复抽取。



数据的随机选取

- 为什么要用Bootstrap抽样?

1. **降低决策树间的相关性**：自助采样法通过为每棵树生成不同的训练集，保证了每棵树都有不同的视角来看待数据，减少了树与树之间的相关性，从而提高了泛化能力。



数据的随机选取

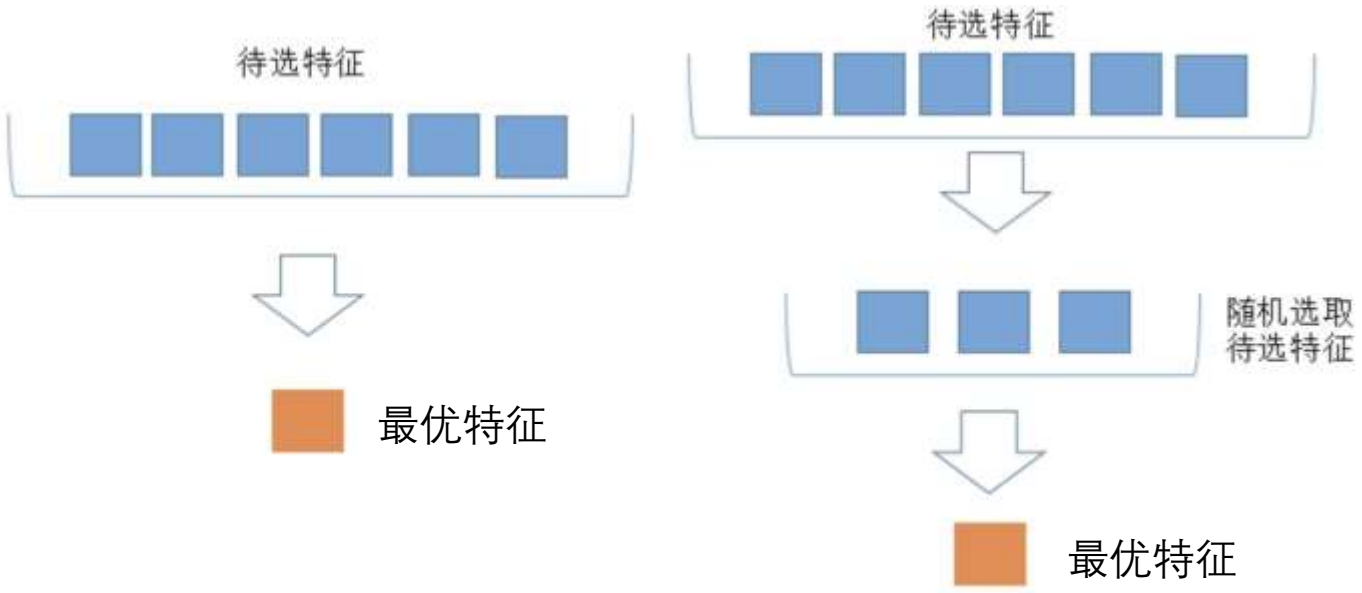
- 为什么要用Bootstrap抽样?

2. **包外估计**：在自助采样法中，那些从未抽到的样本大约是原始数据集的36.8%，可以用来估计模型的泛化误差，而不需要准备额外的测试集。我们把这部分数据称为“包外”（Out-of-Bag, OOB）数据。

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=\frac{1}{e}\approx 0.3679$$

节点特征的随机

- **随机选择属性子集：**在随机森林中，每个决策树在划分节点时，先从所有待选特征中随机抽取 k 个特征作为候选子集，然后再从该子集中选出最优特征进行节点划分。



这一机制促使随机森林在训练过程中探索不同的特征组合和多样化的树结构。

节点特征的随机

在随机森林中，**选择 k（每个节点随机考虑的特征数量）**的推荐通常遵循经验法则，具体取决于问题类型：

1. 分类问题

常用经验法则：

$$k = \sqrt{M}$$

其中 M是总特征数。

2. 回归问题

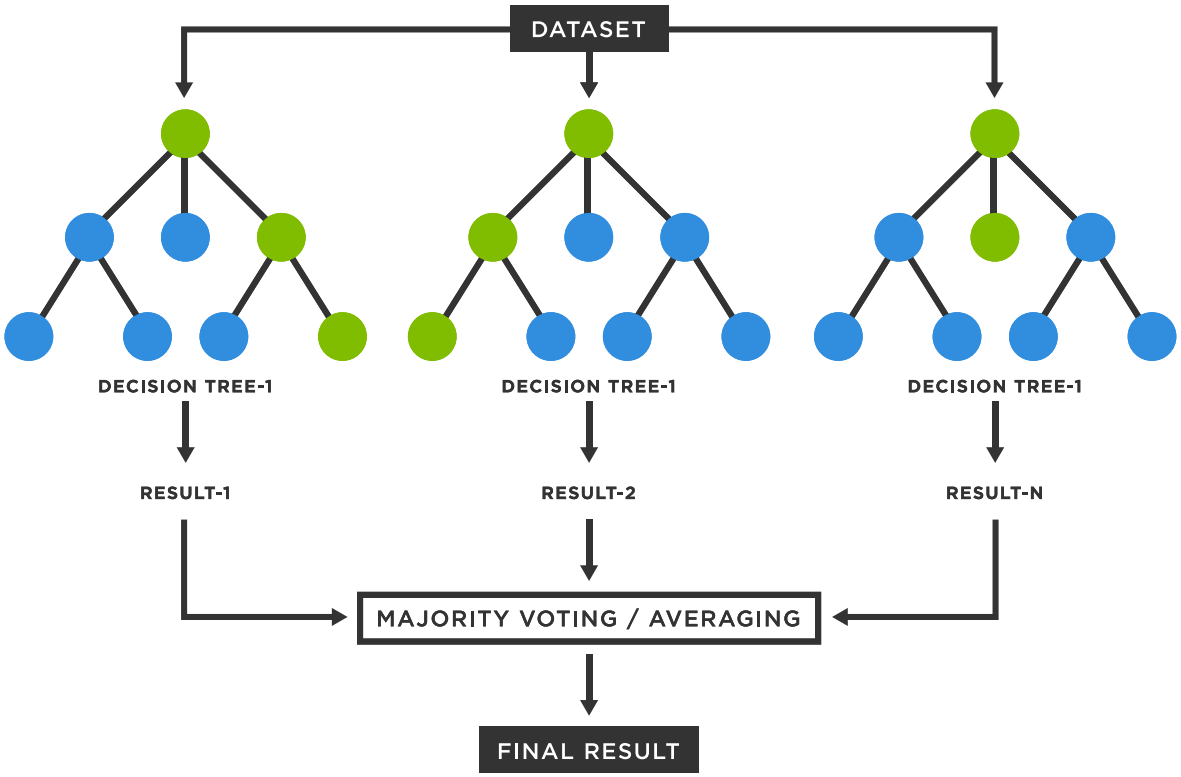
常用经验法则：

$$k = \frac{M}{3}$$

3. 对于非常高维的数据（如文本或基因数据），有时会用 $k = \log_2(M)$或固定常数（如 10~50）来进一步增加随机性。

随机森林的特点分析

- 随机森林通过**在多个决策树上分布计算任务**，能够提高处理大型数据集的效率。（随机森林具有天然的并行性，可以在多个处理器或分布式环境中同时进行，从而显著提高训练速度。）

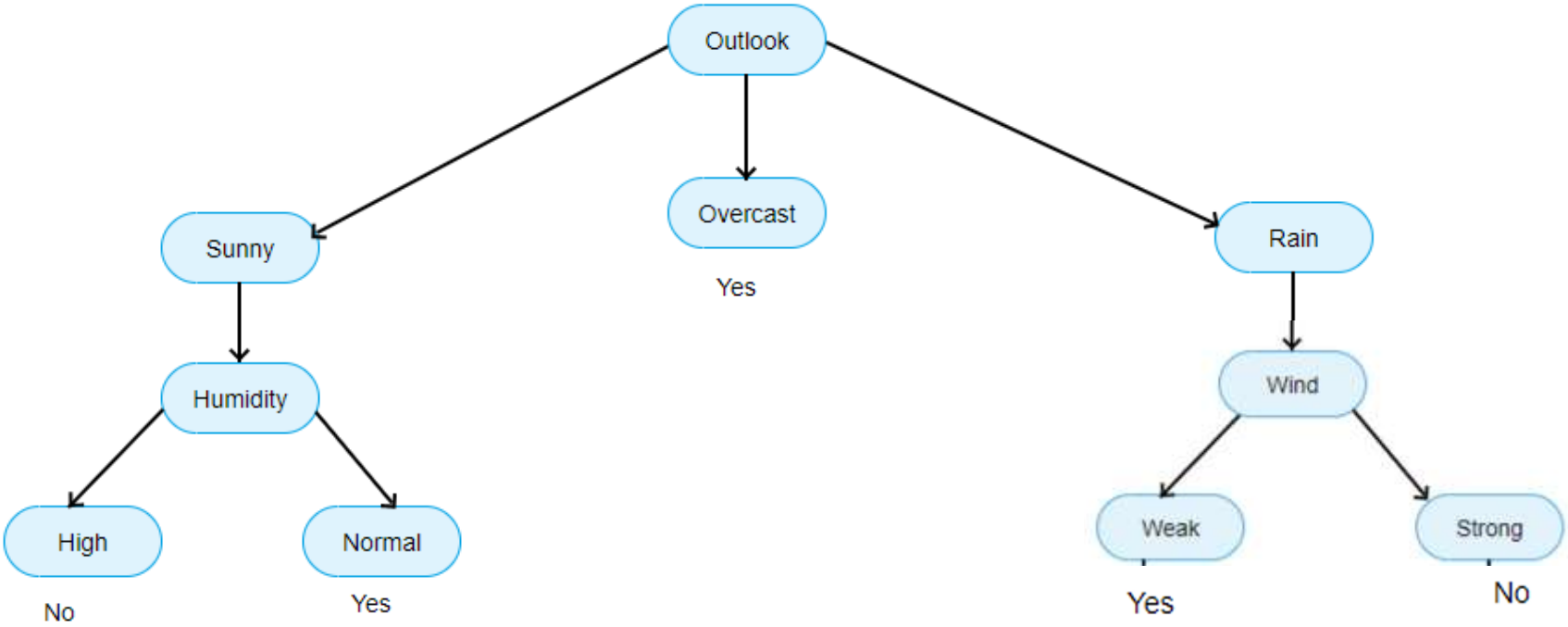


并行的构建树以及并行的推断（分类/回归）

随机森林的特点分析

能评估并确定各特征对分类的重要性。

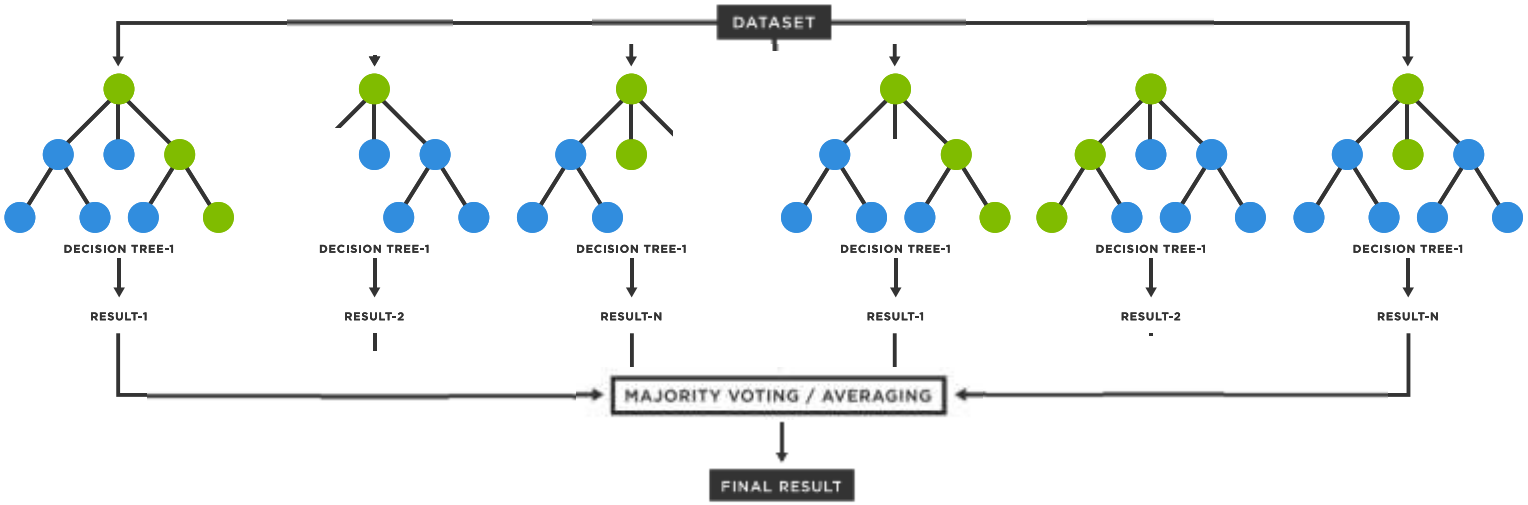
- 在每棵决策树中，当一个特征被用来分裂节点时，会使该节点的纯度提高（例如**降低基尼指数**或**信息熵**）。随机森林统计每个特征在所有树中引起的不纯度降低的总量或平均值，这个量就代表了该特征在整个模型中的贡献和重要性。



随机森林的特点分析

数据部分缺失的情况下也能保持一定的准确性

- 训练阶段，在每次节点分裂时，随机森林会从所有特征中随机抽取一部分候选特征进行选择。如果某个特征存在缺失值，而该特征未被选中进行节点分裂，那么缺失值就不会影响该节点的训练结果。
- 预测阶段，如果遇到缺失值，缺失特征可能导致部分树的分裂受限。但由于随机森林中的树是独立的，仍有部分树未使用该缺失特征，因此它们可以继续输出有效结果。最终，通过聚合这些有效树的结果，模型仍能做出准确的预测。



课堂练习

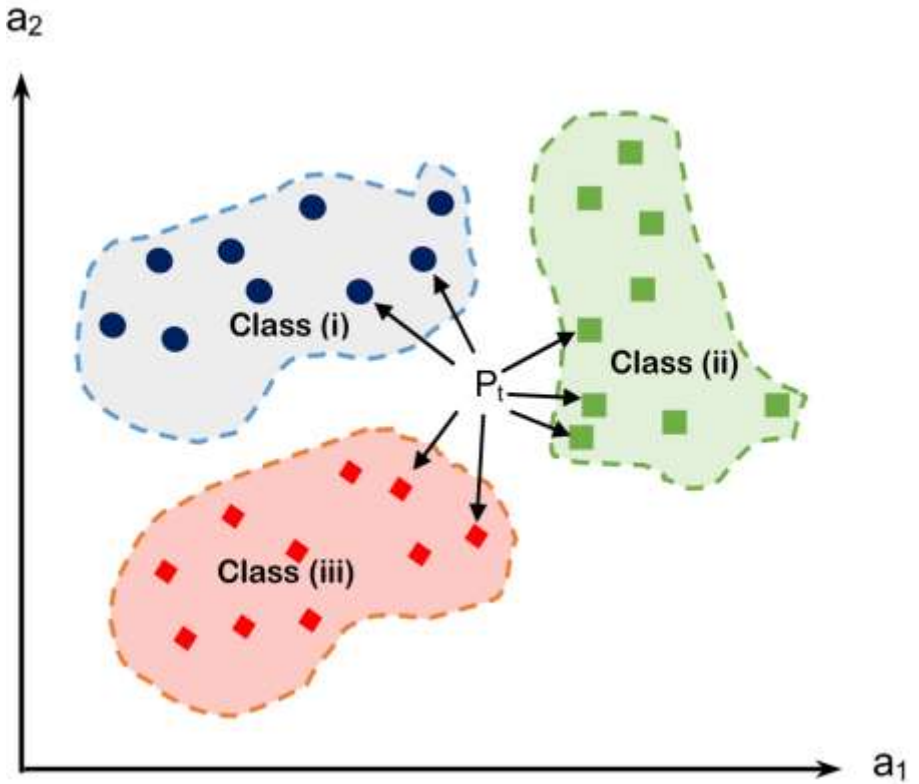
判断以下问题的对/错

1. 随机森林通过构建一个决策树来减少模型的方差。
2. 在随机森林的构建过程中，每棵树的训练数据是从原始数据集中通过有放回抽样（bootstrap sampling）得到的。
3. 随机森林模型中的所有决策树在节点分裂时会考虑所有可用的特征。
4. 随机森林模型通常比单一决策树模型具有更高的计算成本，因为它需要训练多棵树。
5. 随机森林能够减小单一决策树模型方差，但不能减小偏差。

K-近邻算法

K-近邻算法（k-nearest neighbors）

- K近邻算法（k-NN，k-Nearest Neighbors）的基本思想是：对于待预测样本，通过计算其与训练集中所有样本的距离，选取最近的 k 个邻居，然后根据邻居的类别进行分类，或根据邻居的数值进行回归。



问题：点P_t属于哪个类别?

$k = 7 \quad \Rightarrow \quad P_t \in$ 类别(ii)

K-近邻算法

当需要对一个新的样本进行分类时，k-近邻算法会根据某种距离度量，在特征空间中找到最接近该样本的 k 个已知样本（即“邻居”）。在分类任务中，算法通常通过**多数投票法**决定新样本的类别；在回归任务中，算法则通过**邻居的平均值**来预测新样本的值。。

特点：

- **简单，易实现**
- **无参数**
- **懒惰学习**

三要素：

- **距离度量**
- ***k*值的选择**
- **决策规则**

距离度量

- 在数据科学中，特征空间中两个实例点之间的距离反映了它们的相似度。距离越小，表示这两个实例在特征空间中越相似。

- 距离的度量有多种方式。

1. **欧几里得距离 (Euclidean Distance)**
   这是最常见的距离度量，适用于 **数值型** 特征。

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

距离度量

2. 曼哈顿距离 (Manhattan Distance)

计算的是在一个网格坐标系统中，从一个点到另一个点的水平和垂直距离之和。

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

3. 切比雪夫距离 (Chebyshev Distance)

度量的是在各维度上差异的最大值。

$$d(x,y) = \max_i(|x_i - y_i|)$$

距离度量

4. 余弦相似度 (Cosine Similarity)

适用于 **文本数据** 或 **稀疏数据**，用于计算两个向量之间的夹角大小，衡量它们的方向相似度而非大小。

$$\text{cosine similarity}(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$$

其中 $x \cdot y$是向量的点积，‖ x ‖，‖ y ‖是向量的模长。

5. Wasserstein距离

衡量两个概率分布之间差异的距离度量，常用于评估两种分布在某些数据集上的相似度。其原理可以用作“**运输问题**”，也就是说，Wasserstein距离衡量了从一个分布到另一个分布所需的最小“工作量”。

距离度量

6. 汉明距离 (Hamming Distance)

适用于 **二值数据** 或 **离散型数据**，衡量两个等长字符串或序列之间不同的位置数。

$$d(x,y) = \sum_{i=1}^{n} \mathbf{1}(x_i \neq y_i)$$

其中 $\mathbf{1}(x_i \neq y_i)$是指示函数，表示在该位置上 x_i和 y_i是否不相等。

7. 杰卡德相似系数 (Jaccard Similarity)

适用于 **集合** 类型数据，用来衡量两个集合的**距离**。

$$J(x,y) = \frac{|x \cap y|}{|x \cup y|}$$

距离度量

8. **Minkowski 距离（闵可夫斯基距离，L_p 距离）** 是一种 **广义的距离度量**，它把常见的欧氏距离、曼哈顿距离都包含进去了。

给定两个点 $x = (x_1, x_2, ..., x_n)$和 $y = (y_1, y_2, ..., y_n)$，Minkowski 距离定义为：

$$D_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, \quad p \geq 1$$

其中 p是一个可调的参数。

当$p = 1$，称为曼哈顿距离 $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$

当$p = 2$，称为欧式距离 $D_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

当$p = \infty$，称为切比雪夫距离 $D_\infty(x, y) = \max_i |x_i - y_i|$

距离度量

$$D_p(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p},\quad p\geq 1$$

已知二维空间的3个点$x_1=(1,1), x_2=(5,1), x_3=(4,4)$，试求在$p$分别取1，2，3，4时，$L_p$距离下$x_1$的最近邻点。

当$p=1$时	当$p=2$时	当$p=3$时	当$p=4$时
$L_1(x_1,x_2)=$	$L_2(x_1,x_2)=$	$L_3(x_1,x_2)=$	$L_4(x_1,x_2)=$
$L_1(x_1,x_3)=$	$L_2(x_1,x_3)=$	$L_3(x_1,x_3)=$	$L_4(x_1,x_3)=$
L_1距离下x_1的最近邻点是（***）	L_2距离下x_1的最近邻点（***）	L_3距离下x_1的最近邻点（***）	L_4距离下x_1的最近邻点（***）

距离度量

$$D_p(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p},\quad p\geq 1$$

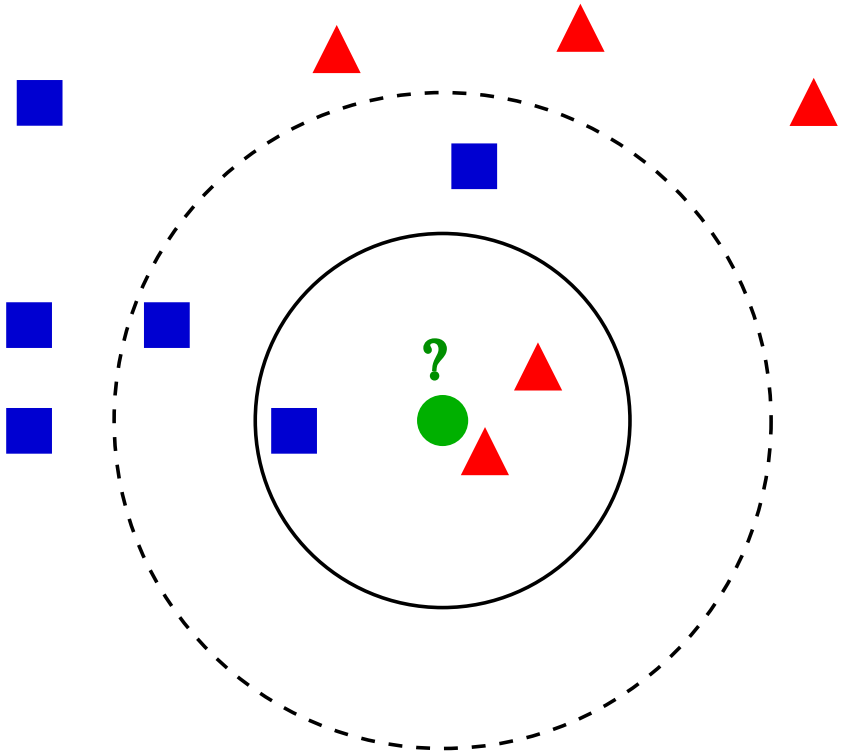
已知二维空间的3个点$x_1=(1,1), x_2=(5,1), x_3=(4,4)$，试求在$p$分别取1，2，3，4时，$L_p$距离下$x_1$的最近邻点。

当$p=1$时	当$p=2$时	当$p=3$时	当$p=4$时
$L_1(x_1,x_2)=4$	$L_2(x_1,x_2)=4$	$L_3(x_1,x_2)=4$	$L_4(x_1,x_2)=4$
$L_1(x_1,x_3)=6$	$L_2(x_1,x_3)=4.24$	$L_3(x_1,x_3)=3.78$	$L_4(x_1,x_3)=3.57$
L_1距离下x_1的最近邻点x_2	L_2距离下x_1的最近邻点x_2	L_3距离下x_1的最近邻点x_3	L_4距离下x_1的最近邻点x_3

*由不同的距离度量所确定的最近邻点是不同的

K值选择

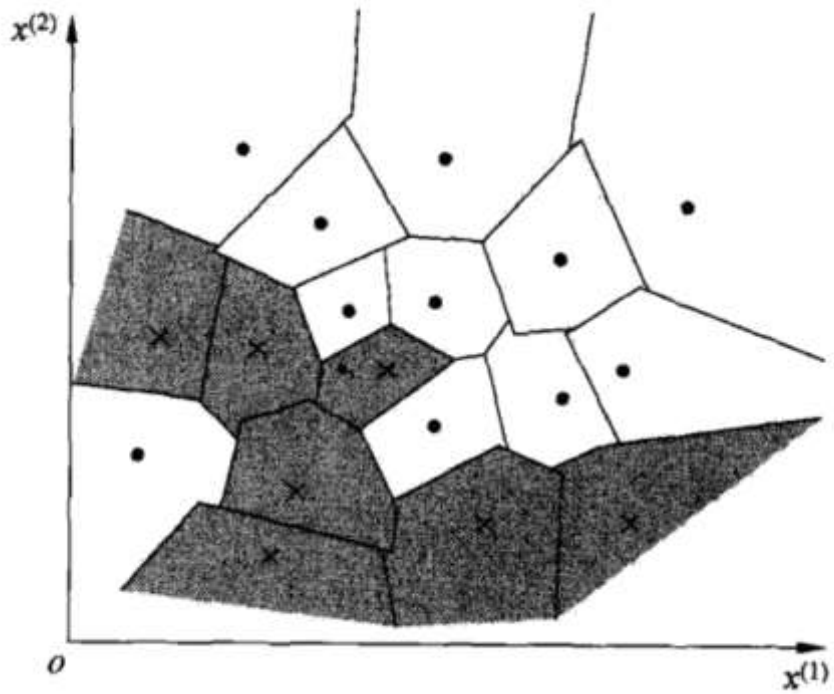
- 不同K值会对结果造成影响



*k*近邻算法例子。如果k=3？如果k=5？

K值选择

- 当**距离度量**、***k* 值**和**决策规则**确定时，K近邻算法可以看作将特征空间隐式划分成若干影响区域（类似子空间）。新样本落入哪个区域，其类别由该区域内的训练样本的投票结果决定（分类问题）。



当k=1（最近邻）时，二维特征空间被分割的例子

K值选择

- **较小的 k 值**？K近邻算法的决策边界高度依赖训练数据的局部样本，特征空间被划分成许多小的影响区域（类似子空间），模型复杂，**容易对训练数据过拟合**。
- **较大的 k值**？每个样本的类别由更多邻居投票决定，决策边界更加平滑，特征空间的影响区域较少且较大，模型简单，但可能无法捕捉数据细节，**容易欠拟合**。

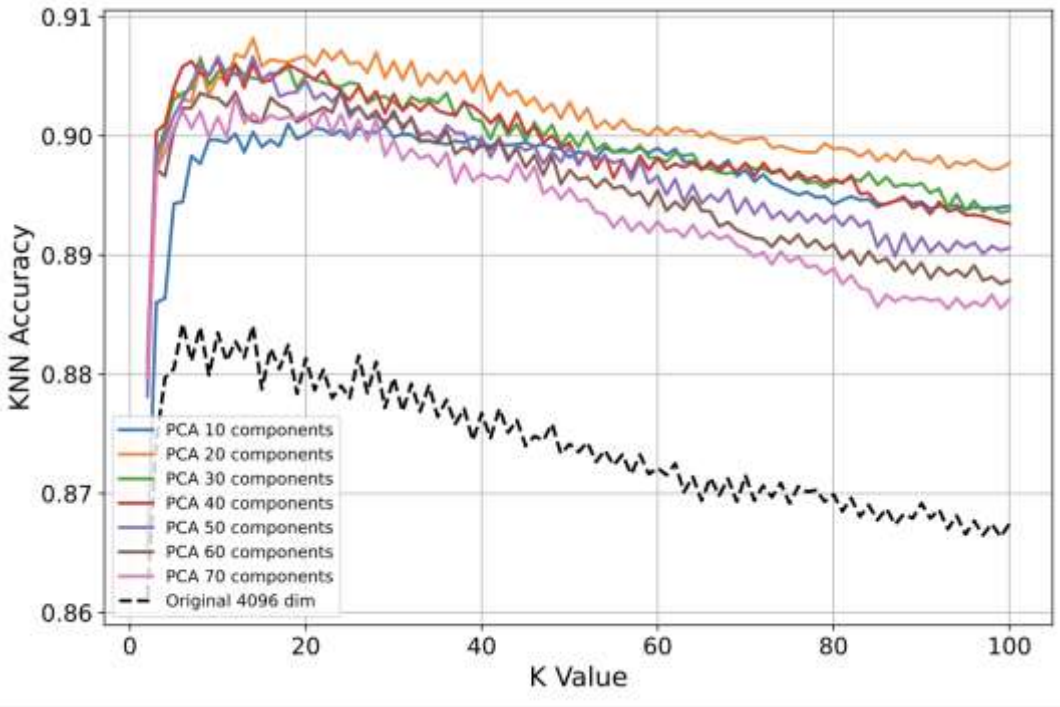
那么K该怎么选呢？

- **经验法则**：有时会用 $k \approx \sqrt{N}$（N 为训练样本数）作为初始参考，再微调。
- **奇偶性原则（分类问题）**通常选择 **奇数** k，避免二分类时出现平票情况。
- 最优 k依赖数据本身特性，**通过交叉验证调整最可靠**。

K值选择

肘部法

1.对不同 k值训练 KNN 模型。
2.在验证集上计算误差率或准确率。
3.绘制 **k vs 验证误差** 曲线。
4.找到误差下降幅度明显减缓的“肘部”，对应的 k即为最佳折中值。



决策规则

在找到K个邻居后如何做决策?

- **分类问题：**

多数投票法（Majority Voting）：在分类问题中，KNN通过查看K个邻居的类别，选择出现最多的类别作为预测结果。

- **回归问题**

均值（或加权均值）法：在回归问题中，KNN的预测结果是K个邻居的标签（值）的平均值。

处理平票情况：

1. 方法1：选择这些类别中距离最近的邻居的类别。
2. 方法2：随机选择一个类别或考虑其他k值。

决策规则

加权KNN（Weighted K-Nearest Neighbors）是KNN的扩展，其中通过对邻居点赋予不同的权重来进行预测，而不是简单的多数投票或均值法。常见的加权KNN方法有：

1. 距离加权KNN（Distance Weighted KNN）

根据每个邻居与查询点的距离来加权，距离较近的邻居对预测结果的贡献较大，距离较远的邻居贡献较小。

通常使用距离的倒数作为权重。对于每个邻居，其权重为：

$$w_i = \frac{1}{d_i}$$

其中，d_i 是第i个邻居与查询点的距离。

决策规则

2. 指数加权KNN（Exponential Weighted KNN）

除了使用距离的倒数外，还可以采用指数函数来加权，通常距离越近的点会获得更高的权重。

$$w_i = e^{-\alpha d_i}$$

其中，α 是一个常数，d_i 是第i个邻居与查询点的距离。每个邻居的贡献依赖于其距离的指数衰减。

优点：可以更灵活地控制权重衰减的速度，适用于更复杂的数据分布。

K-近邻算法的缺点

- **K-NN算法的计算效率低**：

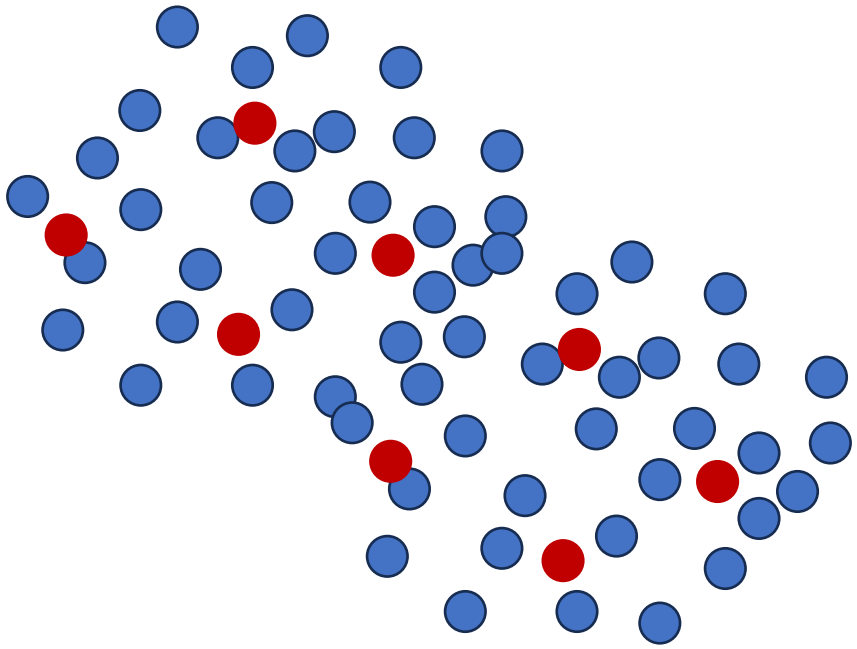
1. **遍历整个训练集：**
   对于每个新样本，算法需要计算它与训练集中所有样本之间的距离，当训练数据量很大时，计算量会非常庞大。
2. **高维数据问题：**
   在高维空间中，距离计算不仅代价高，而且因为维度高，使得距离度量失去区分性，进一步增加了计算负担。
3. **缺乏模型简化：**
   与其他需要预训练模型的算法不同，K-NN没有显式的模型构建过程，所有工作都在预测阶段完成，这意味着无法通过模型压缩来降低预测时的计算复杂度。

Faiss（Facebook AI Similarity Search）库主要是为 高维向量的近似最近邻搜索（KNN） 设计的，在大规模数据集上非常高效。

K-近邻算法的缺点

- **不适用于不平衡的数据集**

在不平衡的数据集中，少数类样本周围很可能被多数类样本包围，使得少数类样本的类别信息被“淹没”，而多数类样本占主导地位，所以KNN会更倾向于预测为多数类。



K-近邻算法的缺点

- **特征比例影响**:

k-NN是基于距离的算法，如果一个特征的范围比其他特征大得多，那么它可能会对距离计算产生不成比例的影响，从而影响算法的性能。因此，在应用k-NN之前一般要进行特征缩放（如标准化或归一化）。

举个例子，假设有一个信用评级数据集，其中包含两个特征:

- **年龄**：范围大约在20到70岁之间
- **年收入**：范围可能在50,000人民币到500,000人民币之间

在这种情况下，由于年收入的取值范围远大于年龄，其差异在距离计算中会起到主导作用。即使两个样本在年龄上差异明显，但只要它们的年收入接近，它们之间的距离也可能很小，就会影响到k-NN的分类效果。

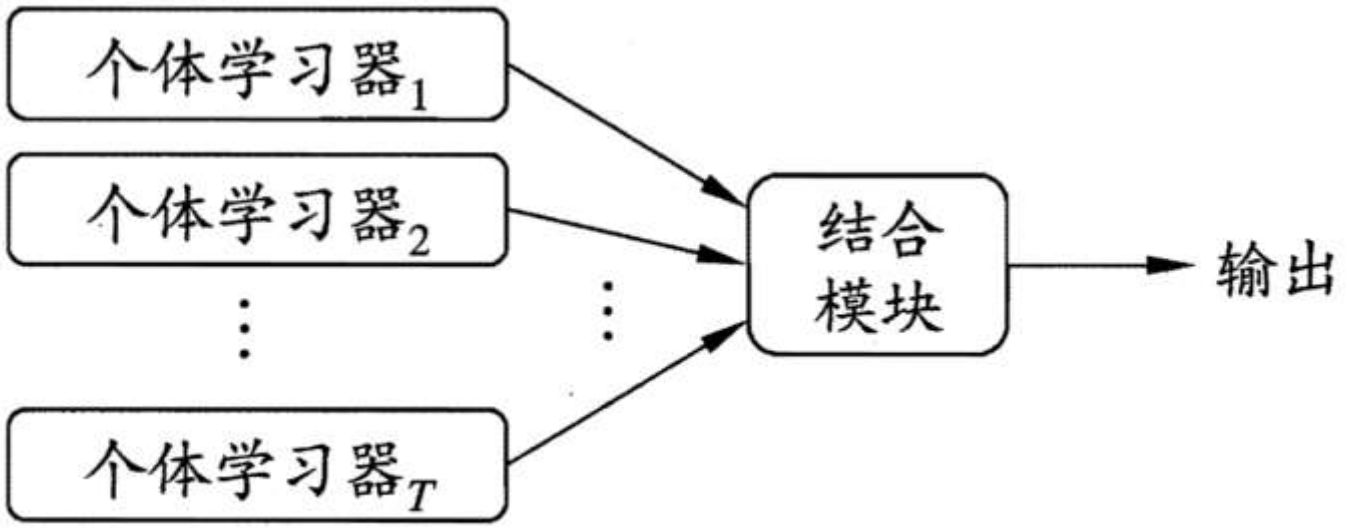
模型组合器
(集成学习)

*一种 **集成学习框架 / 思想**，而不是单一的具体算法

集成学习

- 单一模型（比如决策树）容易 **过拟合 / 不稳定**。
- 我们希望通过“集体智慧”来减少误差。

- 集成学习（Ensemble Learning）通过结合多个学习器（通常称为“个体学习器”或“基学习器”）来提高整体的学习性能，减少误差。



集成学习

同质与异质学习器：

- 同质集成：个体学习器是同种类型的算法。例如，随机森林
- 异质集成：个体学习器是不同类型的算法。

基学习器（个体学习器）的选择：

- 基学习器之间应该“**好而不同**”，不同指的是误差之间应当相互独立。

集成学习策略：

- **Bagging：**个体学习器是独立生成的，如随机森林。学习器之间不存在强依赖关系。
- **Boosting：**后一个学习器的构建依赖于前一个学习器的结果。学习器之间存在强依赖关系。

集成学习： Bagging

"Bagging" 是 "Bootstrap Aggregating" 的缩写

Bagging的核心思想：

1. 从原始训练集通过 **自助采样法（bootstrap sampling）** 反复采样，得到若干不同的子训练集。
2. 在每个子训练集上分别训练一个基学习器（可以是决策树、神经网络等）。
3. 在预测时，将这些基学习器的结果进行聚合。

最典型的具体算法就是 **随机森林（Random Forest）**，它就是在 Bagging 的框架下，用决策树作为基学习器，并在特征选择上再加一层随机性。

Random Forest = Bagging + 决策树 + 特征随机性

集成学习：Bagging

为什么Bagging集成学习使用Bootstrap抽样呢？

- 欲得到泛化性能强的集成，集成中的个体学习器应尽可能相互**独立**；

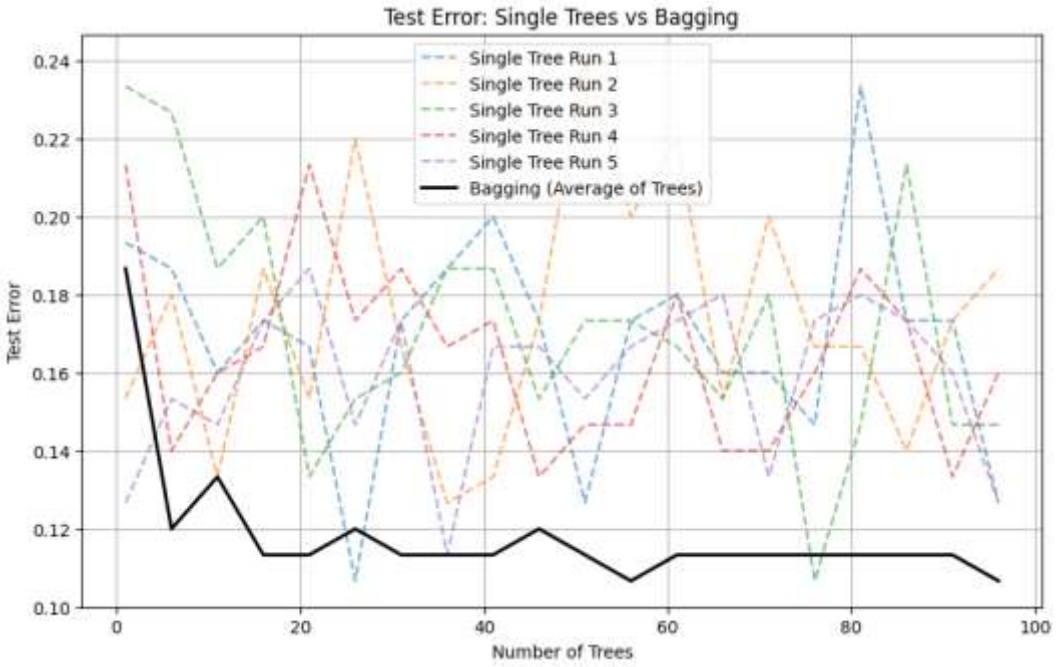
  一般做法：对训练样本进行采样，产生出若干个完全不同的子集，再从每个数据子集中训练出一个基学习器。

  Bootstrap做法：**使用相互有交叠的采样子集（自助采样法）。**

使用bootstrap抽样既能保证数据量的充足，也能保证数据集之间的不同。

集成学习： Bagging

- **对于Bagging这一类算法，**基学习器的误差相互独立时，误差可以相互抵消，可以显著降低 **方差**，但对 **偏差** 的改善有限。

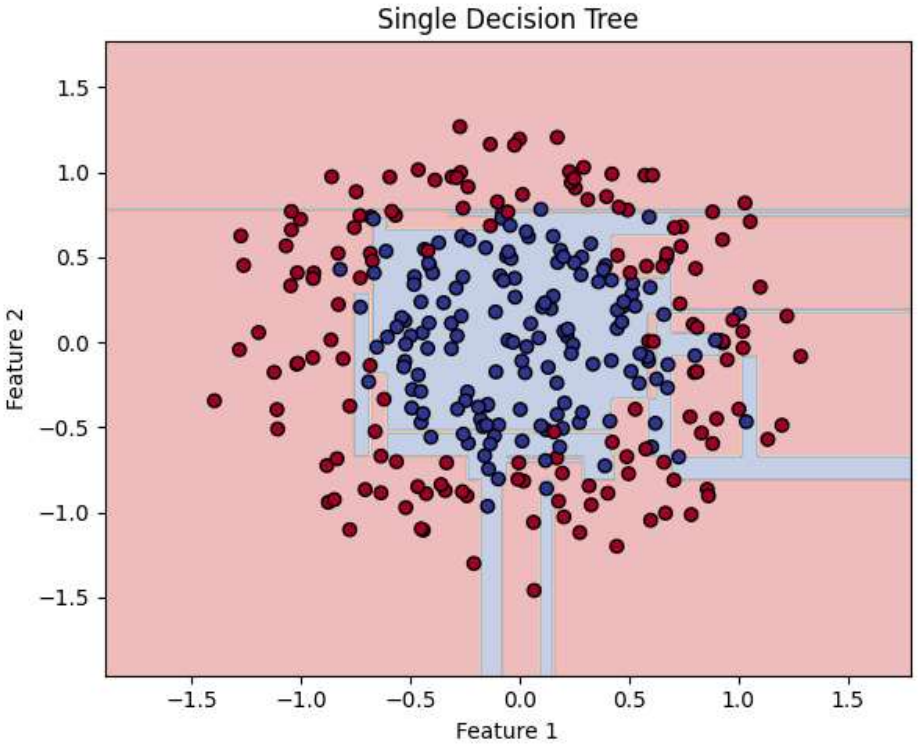


单课树 vs Bagging 误差曲线

- 单课树波动大、结果不稳定（**方差大**）
- Bagging 通过 **平均** 把抖动抵消掉 → **降低方差，**从而平滑，稳定。

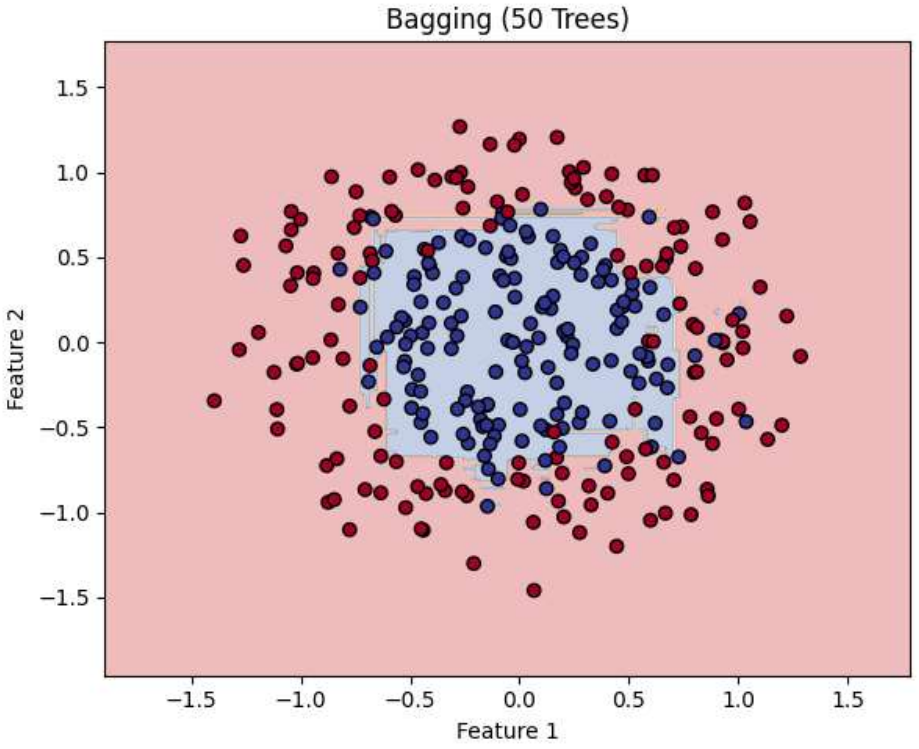
集成学习： Bagging

单棵决策树



边界非常锯齿状、抖动，树对局部数据特别敏感。

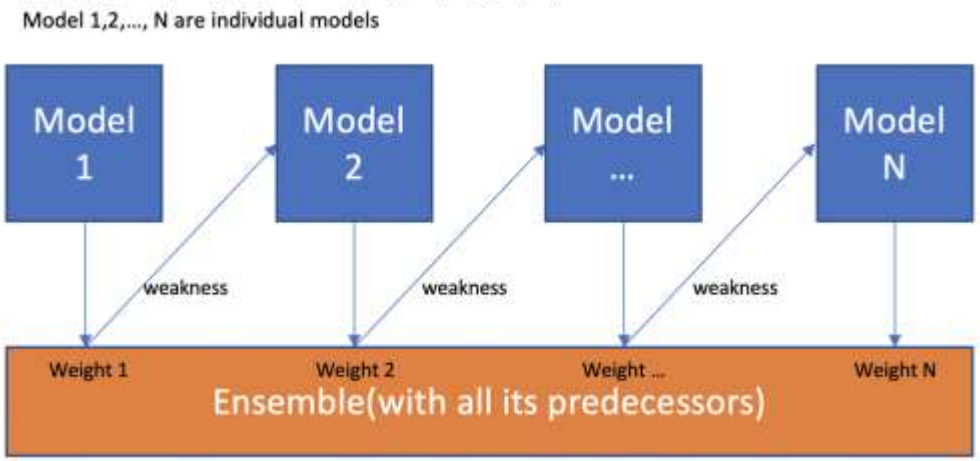
Bagging（50 棵树，bootstrap 采样）



单个树的“随机抖动”会相互抵消，整体边界更平滑。边界更平滑，更接近真实数据分布。

集成学习： Boosting

- **Boosting** 是一种集成学习算法，通过将一系列 **弱学习器**组合起来，构建一个 **强学习器**。每次训练一个基学习器，重点放在 **错误分类的样本** 上，**增加它们的权重**。每个后续学习器都会关注前一个学习器的错误，并通过**调整其权重系数**来改善整体模型的性能。这一过程会持续进行，直到达到预定的基学习器数量或误差无法显著下降。

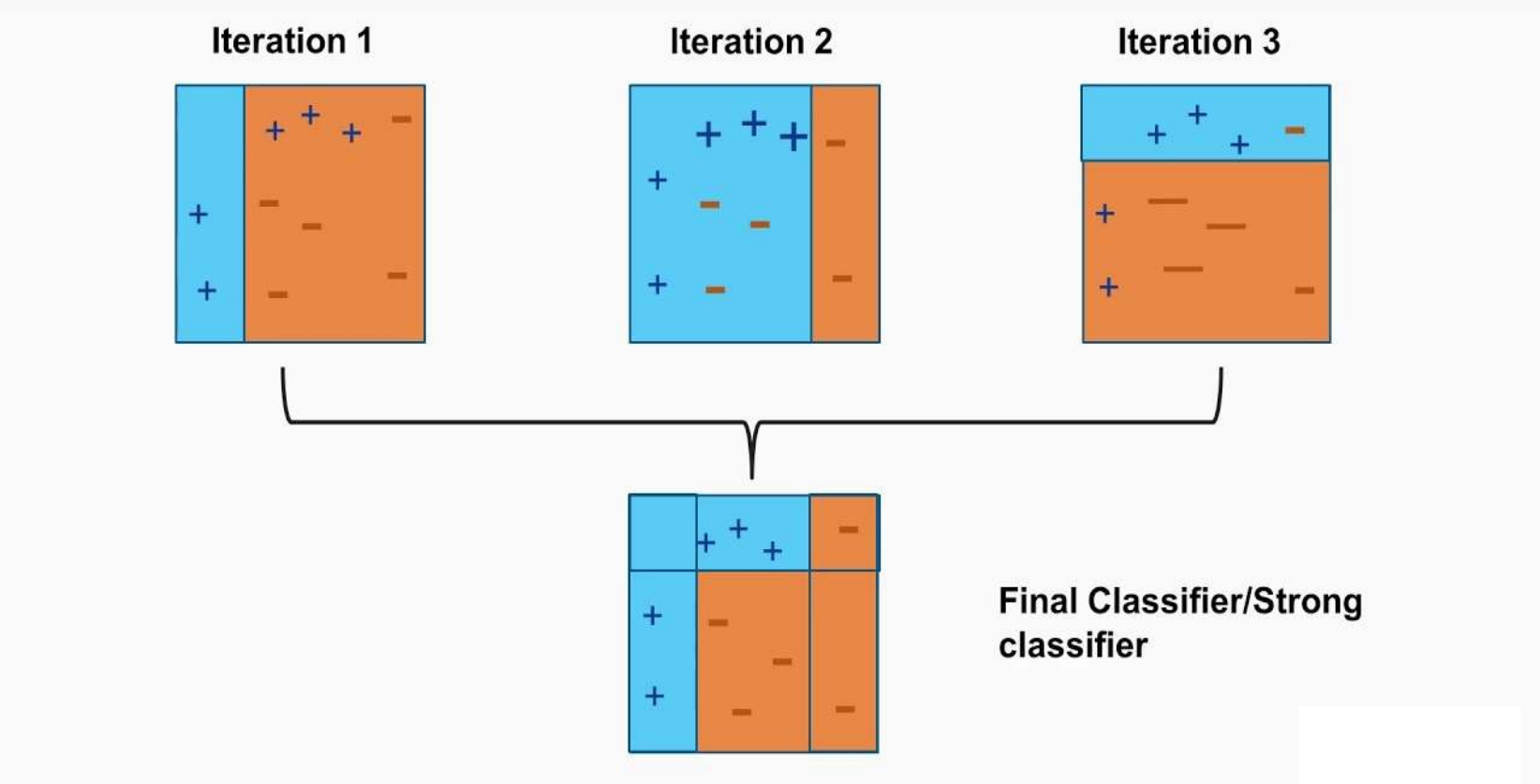


- 大多数 Boosting 方法的弱学习器都是 **决策树** 结构

*弱学习器 是指性能仅 **略好于随机猜测** 的学习器。通常在 **二分类问题** 中，它的准确度略高于50%（例如精度大约在 60% 左右）。**强学习器** 是指能够给出 **接近真实值的准确预测** 的学习器

AdaBoost

- Boosting仅仅是一种框架/思想，具体实现有很多，不同实现所采用的方法不一样。下面以**AdaBoost**为例：



AdaBoost

- 逐轮训练弱分类器（通常是**决策树桩**，即只有一个分裂的树）。
- 每轮根据前一轮的结果 **调整样本权重**：错分的样本权重升高，正确分类的样本权重降低。
- 最终用加权投票的方式聚合所有弱分类器。

假设训练集为 $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$，其中 $y_i \in \{-1, +1\}$。

（1）初始化样本权重

$$w_i^{(1)} = \frac{1}{m}, \quad i = 1, 2, \ldots, m$$

所有样本一开始权重相等。

AdaBoost

（2）循环迭代 $t = 1, 2, \ldots, T$

1. **训练弱分类器**
   用当前样本权重分布 $w^{(t)}$，训练一个弱分类器 $h_t(x)$。

2. **计算分类误差率**

$$\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} w_i^{(t)}$$

   所有被错分样本的权重之和，即加权错误率。

3. **计算弱分类器的权重**

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$$

   错误率越低，权重 α_t越大。

4. **更新样本权重**
   分类错误: $w_i \leftarrow w_i \cdot \exp(\alpha_t)$ 指数为正 → 权重上升

   分类正确: $w_i \leftarrow w_i \cdot \exp(-\alpha_t)$ 指数为负 → 权重下降

AdaBoost

（3）最终输出

输出是所有弱分类器加权组合：

$$\sum_{t=1}^{T} \alpha_t h_t(x)$$

- AdaBoost 的核心是 **调整样本权重**，让难分的样本逐渐被重视。
- 每个弱分类器在最终模型中的话语权由其精度（α_t）决定。
- 本质是加权投票（Weighted Voting），和随机森林的多数投票不同。

AdaBoost

用户	特征 x（交易金额）	标签 y（是否欺诈）
1	10	正常 (+1)
2	20	正常 (+1)
3	50	欺诈 (-1)
4	60	欺诈 (-1)
5	15	欺诈 (-1)

初始化

每个样本权重相等：

$$w_i = 1/5 = 0.2$$

一开始模型对所有交易一视同仁，没有偏好。

AdaBoost

用户	特征 x（交易金额）	标签 y（是否欺诈）
1	10	正常 (+1)
2	20	正常 (+1)
3	50	欺诈 (-1)
4	60	欺诈 (-1)
5	15	欺诈 (-1)

第 1 轮弱学习器

假设弱学习器的规则：**金额 < 30 → 正常 (+1)，否则 → 欺诈 (-1)**

分类结果：

- 样本1，2，3，4正确，5错误
- 计算分类误差率ϵ_1 （即0.2）
- 计算基学习器权重$\alpha_1 = \frac{1}{2}\ln\frac{1-\epsilon_1}{\epsilon_1}$
- 样本 1、2：预测正确 → 权重乘上 $e^{-\alpha_1}$，下降
- 样本 3、4：预测正确 → 权重乘上 $e^{-\alpha_1}$，下降
- 样本 5：预测错误 → 权重乘上 $e^{+\alpha_1}$，上升
- 模型发现第 5 个小额欺诈交易判断错误，AdaBoost 增大其权重，让下一轮学习更关注它。

AdaBoost

用户	特征 x（交易金额）	标签 y（是否欺诈）
1	10	正常 (+1)
2	20	正常 (+1)
3	50	欺诈 (-1)
4	60	欺诈 (-1)
5	15	欺诈 (-1)

第 2 轮弱学习器

重新训练弱学习器时，样本 5 权重变大。样本5的权重和**分类误差率**挂钩，如果样本5在第二轮分类错误，它相对较高权重就会导致误差率变大。为了降低损失，模型会尽力将样本5分类正确。

通过第二轮训练，规则调整为 **金额 < 14 → 正常 (+1)，否则 → 欺诈 (-1)**

分类结果：

- 样本1，3，4，5正确，2错误，样本 5 被正确识别
- 更新权重时：第 5 个样本权重下降（乘上 $e^{-\alpha_2}$），其余正确/错误样本根据分类结果继续调整。
- 模型逐渐“学会”识别那些容易被忽略的欺诈样本。

AdaBoost

用户	特征 x（交易金额）	标签 y（是否欺诈）
1	10	正常 (+1)
2	20	正常 (+1)
3	50	欺诈 (-1)
4	60	欺诈 (-1)
5	15	欺诈 (-1)

后续迭代

每一轮都根据加权错误率 ϵ_t 计算基学习器权重 α_t，再用指数因子 $e^{\pm\alpha_t}$ 调整样本权重，直到弱学习器组合稳定。

最终输出

所有弱分类器加权组合：

$$\sum_{t=1}^{T} \alpha_t h_t(x)$$

集成学习：Boosting

对于Boosting一类算法，因为每个新加入的分类器都专注于前一个分类器错误分类的样本，如果每个分类器都能有效地减少错误，整体错误率将随着分类器数量的增加而指数级下降。

在AdaBoost中，整体模型的错误率由下式给出：

$$\exp\left(-2\sum_{t=1}^{T}\left(\frac{1}{2}-\epsilon_t\right)^2\right)$$

t代表的是分类器的编号，ϵ_t是分类器t的错误率

如果每一个弱分类器的表现都至少比随机猜测好（即每个ϵ_t都小于0.5），那么这个错误率上界会随着分类器数量T的增加而指数级下降。

集成学习：Boosting

常见的booting算法：daBoost (Adaptive Boosting, 1995)、Gradient Boosting (GBDT, 2001)、Stochastic Gradient Boosting (2002)、XGBoost (Extreme Gradient Boosting, 2014)、LightGBM (Light Gradient Boosting Machine, 2016)、CatBoost (2017)

（1）基本思想相同 —— Boosting 框架

- **串行训练弱学习器**（通常是决策树）。
- 每一轮新的学习器都会重点关注 **前一轮没有学好的部分**（错分样本）。

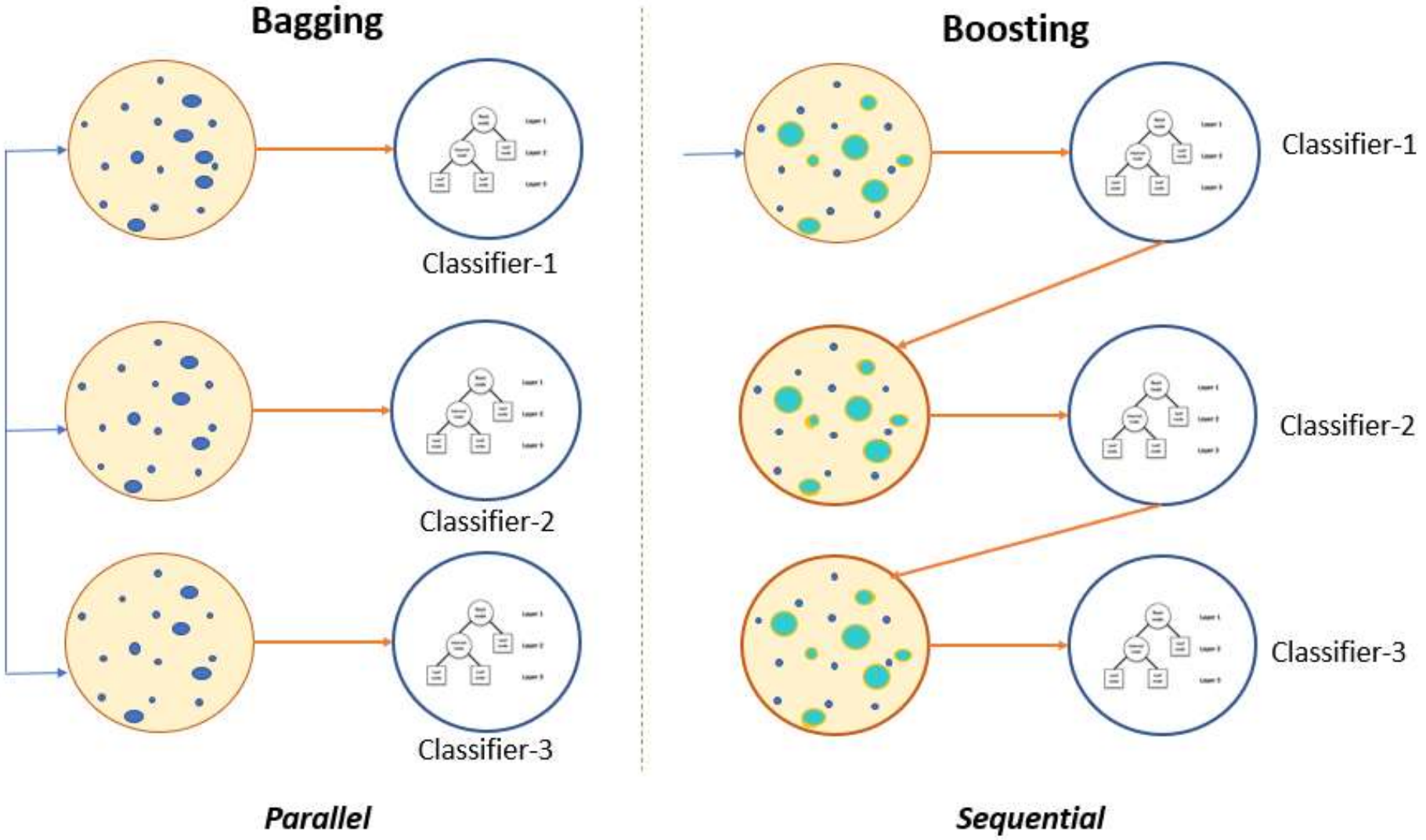
（2）弱学习器相似

- 最常见的弱学习器是 **决策树桩 / 小树**。
- AdaBoost 常用树桩，GBDT/XGBoost/LightGBM/CatBoost 一般用 **深度较小的决策树**。

（3）集成方式相同

- 最终预测都是 **加权和**

Bagging和Boosting的对比



Bagging和Boosting的对比

Bagging 的特点：

- **减少方差：**多个模型的结果将会被平均（回归问题）或者是多数投票（分类问题），这样可以减少模型因数据随机波动造成的方差。
- **防止过拟合：**单个模型可能会在训练集上学得太好，以至于学到了数据中的噪声。当多个这样的模型平均或投票时，噪声的影响会减弱，因为不太可能所有模型都在相同的噪声上过拟合。
- **并行处理：**由于每个分类器是独立的，因此可以并行训练，提高效率。

Boosting 的特点：

- **减少偏差：**通过专注于难以分类的样本，Boosting 能够创建一个强分类器，从而减少模型的偏差。
- **过拟合风险**：如果数据噪声较大，Boosting 方法可能会过分关注错误样本，从而导致过拟合。
- **计算复杂**：由于模型是顺序构建的，不能进行并行处理，这会增加训练时间。

判断题

1. Bagging 是一种只能用于分类任务的集成学习技术。
2. Boosting 方法通过训练多个弱分类器并最终结合它们的预测结果来提高模型的准确性。
3. AdaBoost方法通过增加之前分类错误的样本权重来迫使后续分类器更加关注这些样本。
4. 集成学习方法不适用于处理高维数据。
5. 在集成学习中，如果单个模型过于复杂，则整体模型一定会过拟合。
6. Boosting算法在训练过程中可以并行训练各个分类器。
7. 集成学习中使用的所有模型都必须是同一类型的模型。

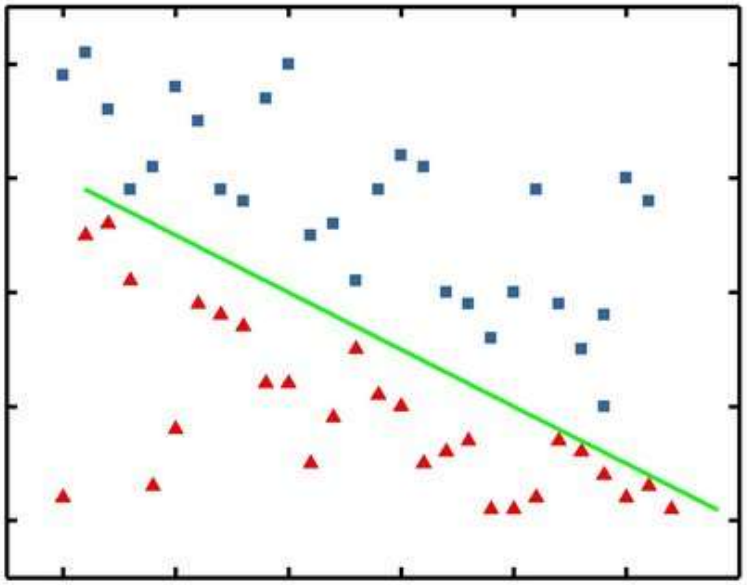
判断题

1. Bagging 是一种只能用于分类任务的集成学习技术。（错误）
2. Boosting 方法通过训练多个弱分类器并最终结合它们的预测结果来提高模型的准确性。（正确）
3. AdaBoost算法通过增加之前分类错误的样本权重来迫使后续分类器更加关注这些样本。（正确）
4. 集成学习方法不适用于处理高维数据。（错误）
5. 在集成学习中，如果单个模型过于复杂，则整体模型一定会过拟合。（错误）
6. Boosting算法在训练过程中可以并行训练各个分类器。（错误）
7. 集成学习中使用的所有模型都必须是同一类型的模型。（错误）

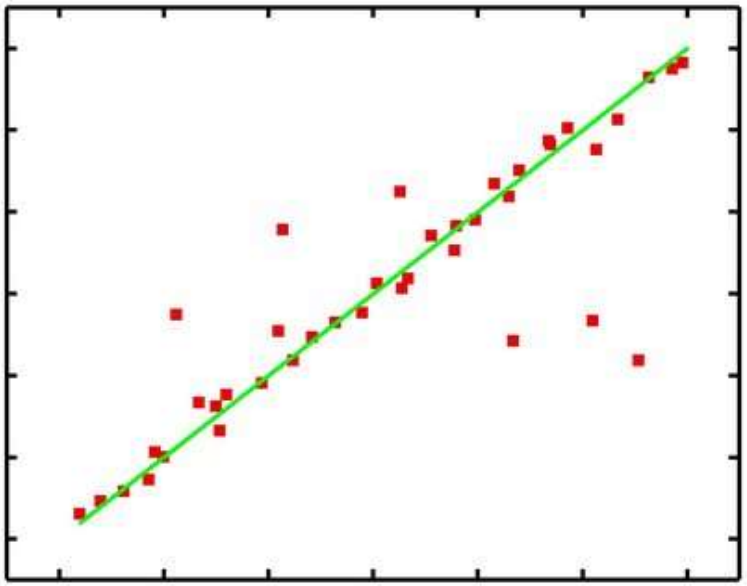
逻辑回归

逻辑回归（Logistic regression）

- 逻辑回归虽名为“回归”，但实际上是用于“分类”问题。“逻辑”源于“Logistic”的翻译，但实际上的logistic指的是Logistic函数（对数几率函数）。
- 逻辑回归的核心是使用Logistic函数（或称对数几率函数）来建模分类问题的概率。因此，逻辑回归有时也被称为**对数几率回归**。
- 逻辑回归与线性回归：线性回归用线性模型预测连续的变量，而逻辑回归用线性模型预测二元结果（是/否，1/0）



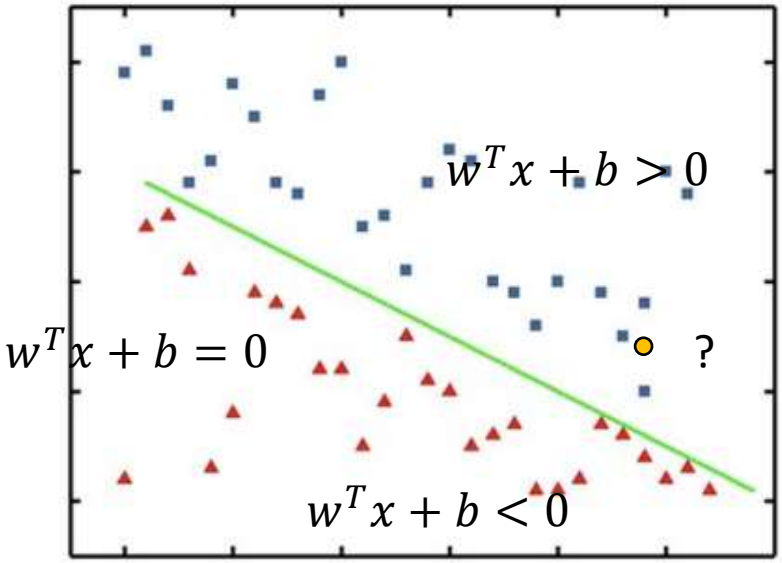
(a) Logistic Regression

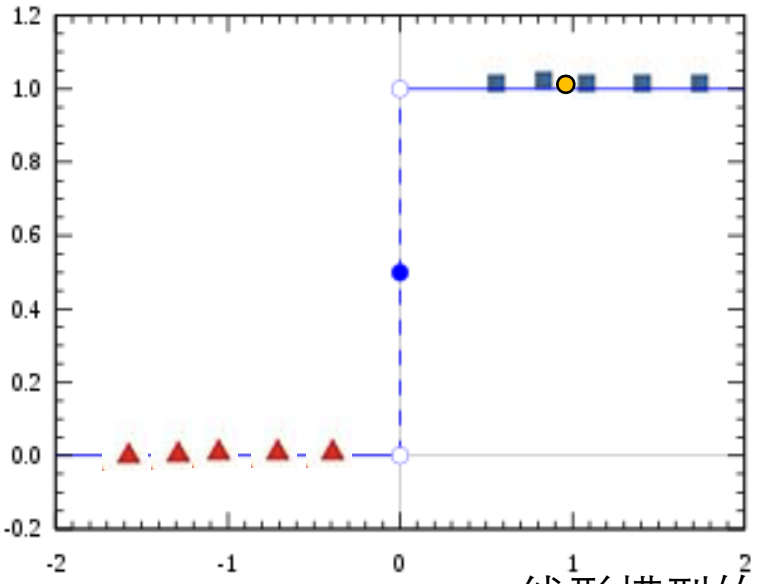


(b) Linear Regression

线性模型的二元分类

- 逻辑回归怎么通过线性模型的输出（连续的值）来进行二分类（离散的值）呢?
  - ✓ 最简单的方式是将线性模型的输出通过单位阶跃函数映射到{0, 1}就可以解决二分类问题。
  - ✓ 这种方法中存在局限性。如果直接将线性回归模型的输出通过单位阶跃函数来映射到{0, 1}，则**缺乏概率解释**。





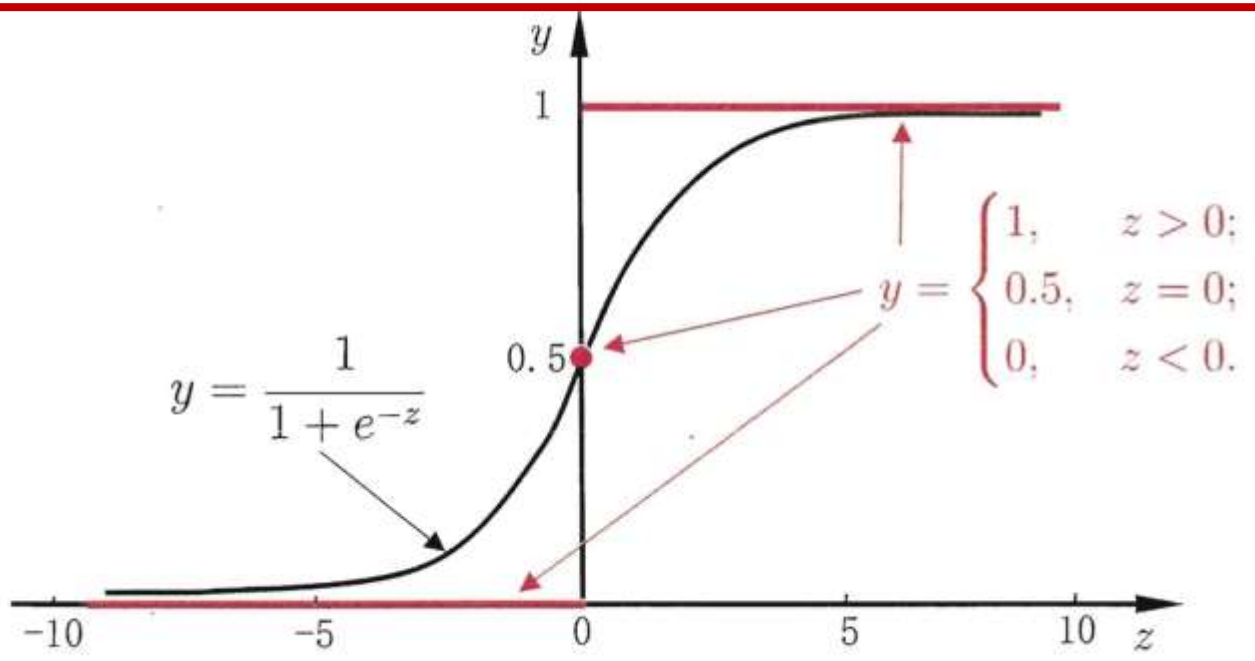
线形模型的输出

单位阶跃函数在 0 处不可导，梯度优化（梯度下降）没法用。

单位阶跃 vs.对数几率

在统计学中，logisctic函数；在机器学习/深度学习中，叫做sigmoid函数

- 将单位阶跃函数替换为对数几率函数（logistic函数）？

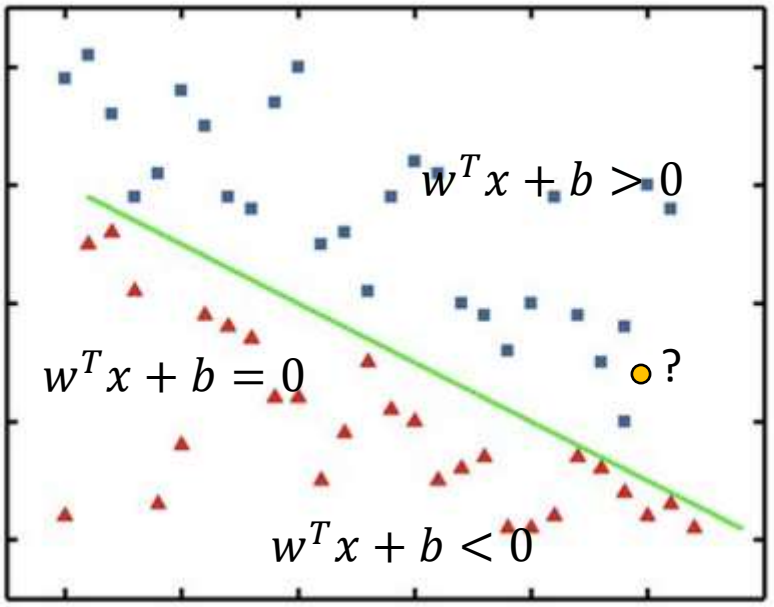


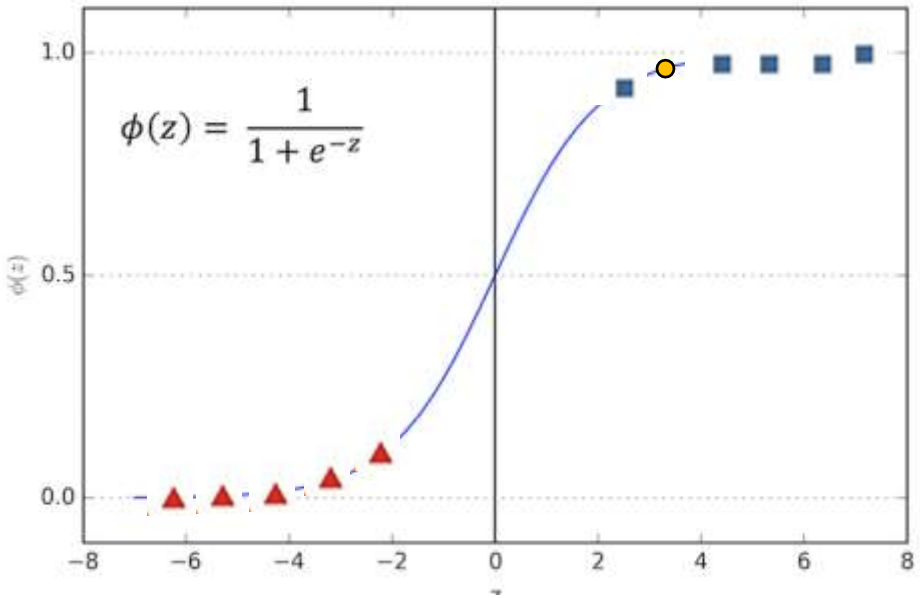
对数几率函数的特性

- logistic函数的输出始终在0和1之间，非常适合表示概率。
- 当z的值增加时，Sigmoid函数逼近1；当z的值减少时，函数值逼近0。函数在$z = 0$附近最为敏感。
- logistic函数是一个平滑的连续函数，可微分，可用梯度法训练（学习）。

逻辑回归（Logistic regression）

- 更为合适的解决方案：用对数几率函数代替单位阶跃函数
- 通过Logistic函数（或对数几率函数）可以将线性模型的输出映射到(0，1)区间内，这样的输出就可以解释为属于某类的概率。Logistic函数将输出限制在0和1之间，同时提供了平滑的概率过渡。





逻辑回归：模型

$$\frac{1}{1+e^{-z}}$$

在逻辑回归中，首先构建一个线性模型：

$$Z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

其中，w_0，w_1，$w, \ldots, w_n$ 是模型参数，$x_1, x_2, \ldots, x_n$ 是特征值。然后，将这个线性模型的输出z作为logistic函数的输入，即：

$$P(y = 1 | (x_1, x_2, \ldots, x_n)) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

$$= \frac{1}{1 + e^{-(w^T x + b)}}$$

样本（$x_1, x_2, \ldots, x_n$）属于**正类别（$y = 1$）**的概率为

$$= \frac{e^{(w^T x + b)}}{1 + e^{(w^T x + b)}}$$

逻辑回归：模型

$$P(y=0|(x_1,x_2,\dots,x_n)) = 1 - \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+\cdots+w_nx_n)}}$$

$$= \frac{1}{1+e^{(w^Tx+b)}}$$

样本（$x_1,x_2,\dots,x_n$）属于**负类别（$y=0$）**的概率为 ➡ $\frac{1}{1+e^{(w^Tx+b)}}$

样本x属于正类别的概率：$p(y=1 \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$

样本x属于负类别的概率：$p(y=0 \mid \boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$.

逻辑回归：策略

- 希望每个样本属于其真实标记的概率越大越好。

（1）如果是正类样本，那就希望**模型预测**它为正类别的概率接近1。

（2）如果是负类样本，那就希望**模型预测**它为负类别的概率接近1。

策略：通过最大化训练集样本的预测概率乘积，选择最优模型参数。

训练集样本x属于正类的概率：$$p(y=1 \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$$

训练集样本x属于负类的概率：$$p(y=0 \mid \boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}} .$$

整个训练集的似然函数，即为：

σ就是指 **logistic 函数，** $y_i \in \{0, 1\}$

$$L(w,b) = \prod_{i=1}^{N} P(y_i|x_i;w,b) = \prod_{i=1}^{N} \left(\sigma(w^{\top}x_i+b)^{y_i}\left[1-\sigma(w^{\top}x_i+b)\right]^{1-y_i}\right)$$

逻辑回归：策略

- 原始乘积容易下溢（数值变得很小），所以优化时通常取对数，将乘积转换为求和，得到对数似然。

  似然函数取对数得到对数似然

$$\ell(w,b) = \log L(w,b) = \sum_{i=1}^{N} \Big[y_i \log \sigma(w^\top x_i + b) + (1-y_i) \log (1 - \sigma(w^\top x_i + b)) \Big]$$

- 把对数似然取负号（偏好求最小值而非最大值）就变成 **损失函数**：

  逻辑回归的优化目标是最小化：

$$J(w,b) = -\ell(w,b) = -\sum_{i=1}^{N} \Big[y_i \log \sigma(w^\top x_i + b) + (1-y_i) \log (1 - \sigma(w^\top x_i + b)) \Big]$$

  这个损失函数也称作**交叉熵损失**。

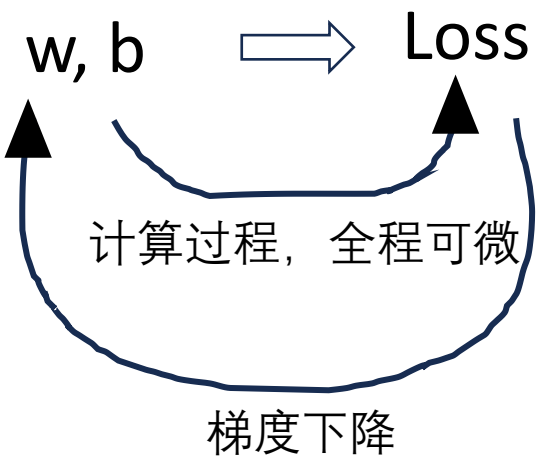
逻辑回归：算法

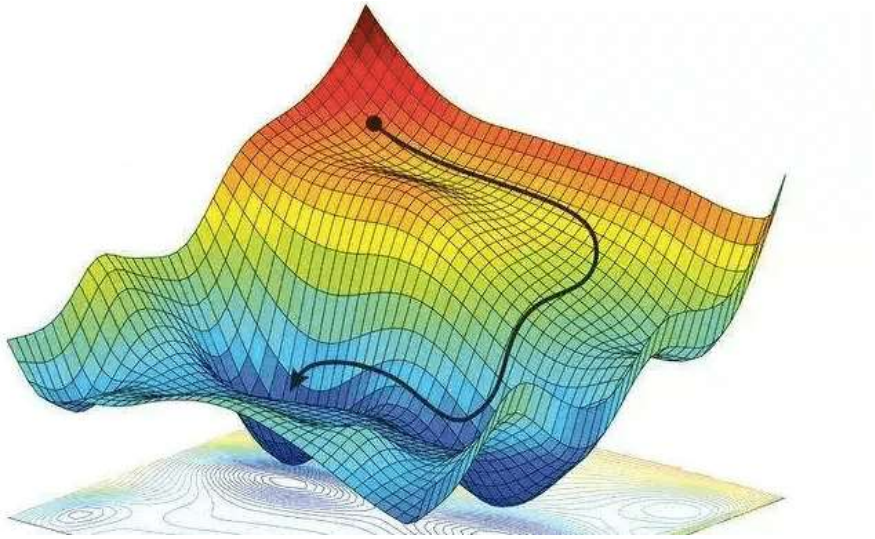
优化目标:

- 最小化

$$J(w,b) = -\ell(w,b) = -\sum_{i=1}^{N}\left[y_i \log \sigma(w^\top x_i + b) + (1-y_i)\log(1-\sigma(w^\top x_i + b))\right]$$

算法：梯度下降

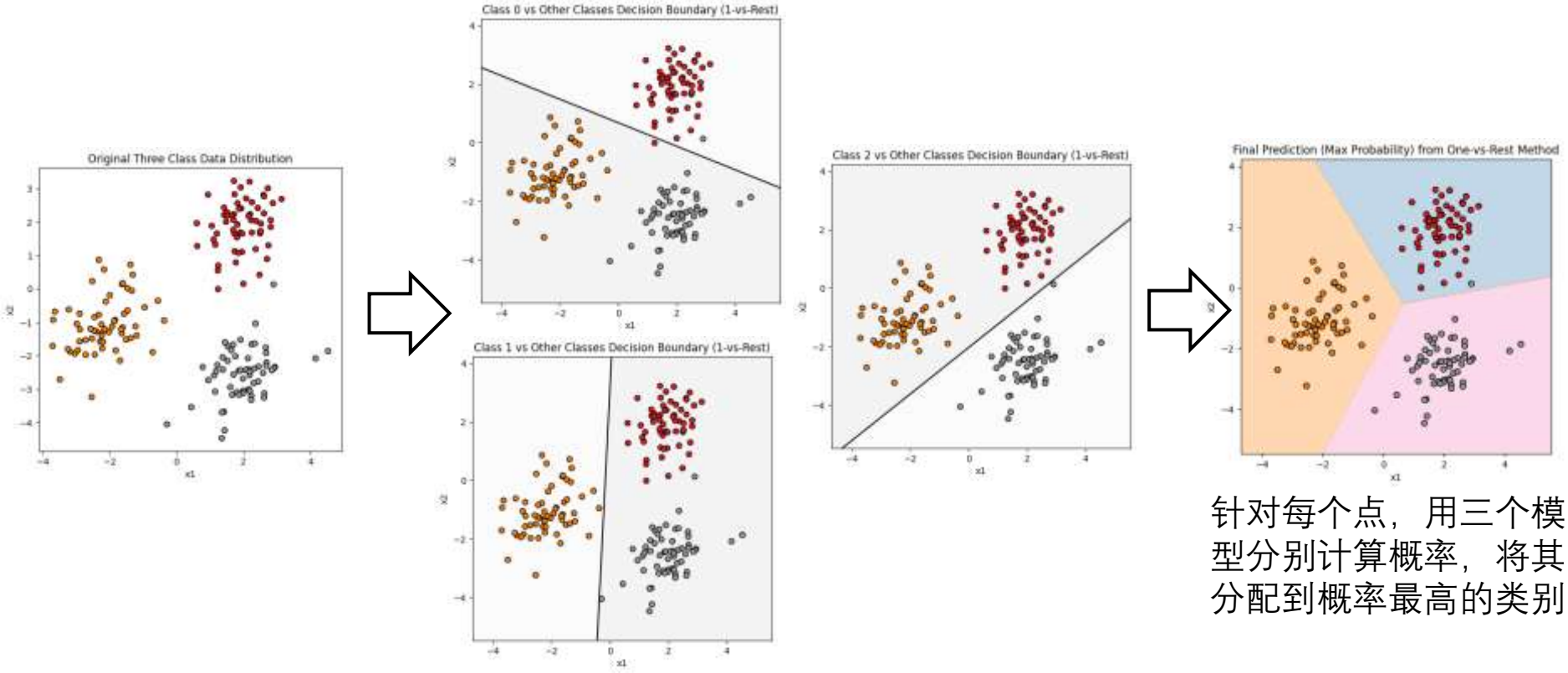




多类别问题怎么办?

逻辑回归原生是二分类的，如果想做 K 类分类：

- 对每一类 k 建立一个二分类逻辑回归，预测时选择 **概率最大的类别**
- 每个类别都有自己的超平面，把这个类别与其他类别分开。



针对每个点，用三个模型分别计算概率，将其分配到概率最高的类别

课题习题

判断对错：

1. 逻辑回归解决的是回归问题。
2. 逻辑回归模型本质上是一个线性模型。
3. 逻辑回归模型的输出值是介于0和1之间的连续值。
4. 标准的逻辑回归只能用于处理二分类问题。
5. 逻辑回归要求自变量和因变量（事件发生概率）之间具有线性关系。

课题习题答案

1.逻辑回归解决的是回归问题。（错误）

2.逻辑回归模型本质上是一个线性模型。（正确，逻辑回归尽管预测的是概率，但其核心是线性的，因为它使用的是自变量的线性组合来预测。

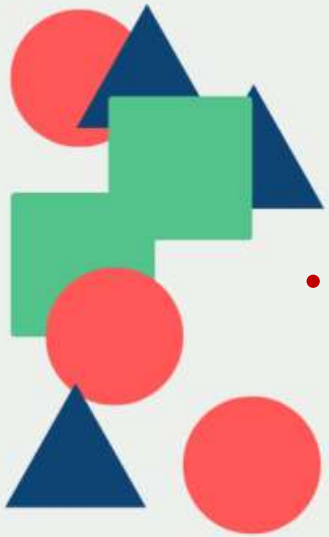
3.逻辑回归模型的输出值是介于0和1之间的连续值。（正确，逻辑回归通过sigmoid函数）

4.标准的逻辑回归只能用于处理二分类问题。（正确）

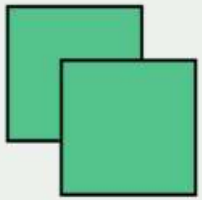
5.逻辑回归要求自变量和因变量（事件发生概率）之间具有线性关系。（错误）

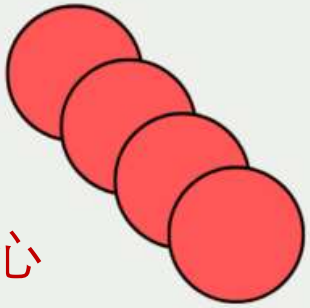
朴素贝叶斯

朴素贝叶斯分类器



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





- **贝叶斯定理是**朴素贝叶斯分类器的核心

应用:

- 垃圾邮件分类
- 文本分类

贝叶斯定理（Bayes' theorem）

- 贝叶斯定理是用来计算在给定 新证据或数据 之后某个事件的概率（**后验概率**）。

$$P(A \mid B) = \frac{P(A)P(B \mid A)}{P(B)}$$

- 例：可以通过癌症的总体平均风险来预测一个人患病的概率（**先验概率**）。使用贝叶斯定理之后，我们就可以进一步结合个体的具体信息（如年龄、性别、遗传因素等）来提供更精确的风险估计（**后验概率**），而不是仅仅依赖于总体平均风险。

先验概率/后验概率

- **先验概率（Prior Probability）**：这是在考虑任何相关证据或数据之前，根据以往经验或主观判断预估某事件发生的概率。

  例：如果某地区过去十年中每年都发生地震，那么我们可能会基于这一经验判断来年也会发生地震的概率较高，这就是先验概率。

- **后验概率（Posterior Probability）**：在观察或获取新的数据后，对事件发生概率的重新评估。

  例：如果在考虑了最近的地质活动，天气情况后，我们发现该地区的地震活动有所下降，那么我们可能会降低来年发生地震的概率评估，这就是后验概率。

分类任务下的先验概率/后验概率

- 先验概率 $P(C)$ 表示在任何特定观测数据X被考虑之前，事件 C（通常是一个类别标签或某种结果）的概率。（比如：给定如下数据集，现在有一个水果，问它属于每个类别的概率（Banana，Orange，Other））

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

- 后验概率$P(C|X)$表示在给定观测数据X（即特征）的条件下，目标变量 C（即类别）的概率。（比如：我进一步给出水果特征为Long，Not sweet，Yellow，问它属于每个类别的概率（Banana，Orange，Other））

- 分类任务中核心的问题：分类问题 ⇒ 在给定观测数据的特征的情况下，预测该数据属于各个类别的概率 ⇒ **后验概率**。

朴素贝叶斯分类

假如我们的分类模型中的某一个样本是:

$$\mathbf{x} = (x_1, \ldots, x_n)$$

- 分类问题则可以转化为求如下后验概率的问题

$$p(C_k \mid x_1, \ldots, x_n)$$

$$P(C_k|X)$$

- 对于给定的特征X，可以通过计算后验概率$p(c|x)$来预测c (类别)

$$\hat{C} = \arg\max_{C_k} P(C_k|X)$$

朴素贝叶斯分类

- **贝叶斯定理可以用来估计后验概率**

根据贝叶斯定理，我们有

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

（后验概率）（先验概率）（似然）

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

（“证据”因子）

朴素贝叶斯分类

先验概率 $P(C_k)$由数据集 D 中类别分布确定，
通常通过统计类别标签的频率来估计。

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

- **先验概率 $P(C_k)$**在 **给定类别 C_k**的情况下是一个 **常数**，因为它仅与数据集 D中各个类别的分布有关，而与特征 X无关。

朴素贝叶斯分类

在类别 C_k的背景下，观测到特征 X的概率。即我们在假设类别为 C_k的情况下，看到样本 X的可能性。

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

- 在样本量充足的情况下，我们通常可以通过频率估计法（即根据数据中各特征出现的频率）来准确估计似然 $P(X \mid C_k)$。
- 然而，在实际应用中，样本量可能不足以精准估计似然，这时我们引入了特征独立性假设来简化模型。朴素贝叶斯模型假设在给定类别 C_k 的情况下，所有特征 $X_1, X_2, \ldots, X_n$ 是相互独立的。基于条件独立的定义，似然可以表示为各个特征条件概率的乘积：

$$P(X_1, X_2, \ldots, X_n | C_k) = P(X_1|C_k) \times P(X_2|C_k) \times \cdots \times P(X_n|C_k)$$

朴素贝叶斯分类

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

证据因子 $P(x)$表示在整个模型（所有类别及其先验分布）下观测到样本 x的概率

- 我们关心的是各类别的后验概率 $P(C_0 \mid x), P(C_1 \mid x), \ldots, P(C_k \mid x)$哪个最大，而不需要精确计算它们的具体数值。由于这些后验概率具有相同的分母（即证据因子 $P(x)$），在比较大小时可以忽略证据因子，无需实际计算。

朴素贝叶斯分类

通过统计类别标签的频率来估计。

转化为各个特征条件概率的乘积

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

不需要计算

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} = \frac{p(C_k)}{p(x)} \prod_{i=1}^{d} p(x_i|C_k)$$

朴素贝叶斯模型

$p(C_k, x_1, \ldots, x_n)$

模型估计的是类别和特征的联合概率分布

$$p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

总结：

- 由给定数据集得到输入输出的联合概率分布模型 $p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$
- 将给定的某个样本x，输入到该联合概率分布模型，得到使后验概率最大的k

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

朴素贝叶斯分类

- 朴素贝叶斯分类器（NBC）是一种**基于贝叶斯定理**的分类方法，其核心假设是**所有特征在给定类别的条件下是相互独立的。** 在分类或预测阶段，给定一个新的输入向量X，NBC通过应用贝叶斯定理，计算并比较在每个可能的输出类别Y下的**后验概率**。最终，**模型将选择具有最大后验概率的类别作为预测结果。**

- **特征在给定类别的条件下是相互独立的：**然而在现实世界的数据中，**特征之间往往是相关的**。例如，在图像识别中，相邻像素之间就有很高的相关性；在文本分类中，某些词的出现可能会影响其他词的出现。朴素贝叶斯算法忽略了这些可能的相关性，这种做法在某种意义上是“过于简化”或“朴素”的。

- 如果特征之间不独立，是不是朴素贝叶斯分类器就无法使用呢？

朴素贝叶斯分类

- 朴素贝叶斯分类器（NBC）是一种**基于贝叶斯定理**的分类方法，其核心假设是**所有特征在给定类别的条件下是相互独立的。** 在分类或预测阶段，给定一个新的输入向量X，NBC通过应用贝叶斯定理，计算并比较在每个可能的输出类别Y下的**后验概率**。最终，**模型将选择具有最大后验概率的类别作为预测结果。**

- **特征在给定类别的条件下是相互独立的：**然而在现实世界的数据中，**特征之间往往是相关的**。例如，在图像识别中，相邻像素之间就有很高的相关性；在文本分类中，某些词的出现可能会影响其他词的出现。朴素贝叶斯算法忽略了这些可能的相关性，这种做法在某种意义上是“过于简化”或“朴素”的。

- 如果特征之间不独立，是不是朴素贝叶斯分类器就无法使用呢？（否，数据集的特征之间相关性越高，朴素贝叶斯的分类效果一般就越差。但不代表朴素贝叶斯无法使用。）

朴素贝叶斯分类器的例子

计算例：

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

- 3个类别： banana, orange, other
- 3个特征: Long/Not long, Sweet/Not sweet, Yellow/Not yellow
- 1000个样本（500个banana, 300个orange, 200个other）

问题：现有一个长形状的（Long）、有甜味的（Sweet）、黄颜色（Yello）的水果，你能猜测出这是什么水果吗？

朴素贝叶斯分类器的例子

根据贝叶斯公式:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

$$p(Banana|\ Long, Sweet, Yellow) = \frac{p(Banana) * p(Long, Sweet, Yellow|Banana)}{p(Long, Sweet, Yellow)}$$

$$p(Orange|\ Long, Sweet, Yellow) = \frac{p(Orange) * p(Long, Sweet, Yellow|Orange)}{p(Long, Sweet, Yellow)}$$

$$p(Other|\ Long, Sweet, Yellow) = \frac{p(Other) * p(Long, Sweet, Yellow|Other)}{p(Long, Sweet, Yellow)}$$

朴素贝叶斯分类器的例子

以Banana为例：

$$p(Banana|\ Long, Sweet, Yellow)$$

$$= \frac{p(Banana) * p(Long, Sweet, Yellow|Banana)}{p(Long, Sweet, Yellow)}$$

$$= \frac{p(Banana) * p(Long|Banana) * p(Sweet|Banana) * p(Yellow|Banana)}{p(Long, Sweet, Yellow)}$$

$$= \frac{0.5 * 0.8 * 0.7 * 0.9}{p(Long, Sweet, Yellow)} = \frac{1}{Z} * 0.252$$

朴素贝叶斯分类器的例子

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

$p(Orange|\ Long, Sweet, Yellow) =?$

$p(Other|\ Long, Sweet, Yellow) = ?$

朴素贝叶斯分类器的例子

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

$$p(Orange|\ Long, Sweet, Yellow) = 0$$

$$p(Other|\ Long, Sweet, Yellow) = \frac{1}{z} * 0.01875$$

所以，当给定一个水果的特征：long, sweet, yellow，那么这个水果更有可能是Banana。

$$p(Banana|\ Long, Sweet, Yellow) > p(Other|\ Long, Sweet, Yellow) > p(Orange|\ \ldots)$$

朴素贝叶斯模型的优点

1. **数据需求不高**：即使在数据量较少的情况下，朴素贝叶斯分类器也能估计出必要的参数。*必要的参数包括类别的先验概率和给定类别下特征的条件概率，这些在数据量较少的情况下，也同样可以获得。*

2. **对缺失数据不敏感**：在朴素贝叶斯中，各个特征在给定类别的条件下是独立的，也就是说每个特征对最终的后验的贡献是单独计算的。如果某个或几个特征缺失，只需在计算联合概率时忽略这些特征，而不必影响其他特征的条件概率计算。

$$p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

计算似然时，可以忽略缺失特征，只乘以那些已知的特征的条件概率 $p(x_i \mid C_k)$

3. **简单到几乎不需要调参：**与许多机器学习算法相比，朴素贝叶斯模型在调参方面几乎可以忽略不计，这使得它非常适合对时间和资源有限的快速原型设计。

朴素贝叶斯模型的缺点

1. **特征独立性假设**：朴素贝叶斯模型假设在给定类别条件下所有特征相互独立，但现实数据中，特征往往存在一定关联性。特征间关联性较强时，这种独立性假设不完全成立，可能导致模型性能下降；而特征关联性较弱时，模型仍能取得较好的效果。

2. **零概率问题**：当某个类别下的某个特征未在训练集中出现过时，会导致该类别的概率估计为零，即零概率问题，这会影响到后验概率的计算和分类结果。

$$p(Orange|\ Long, Sweet, Yellow) = 0$$

这是因为Long这个特征并没有在Orange这个类别里出现过，也就是P(Long|Orange)=0，并最终导致后验概率为0。意味着即使其他证据强烈支持样本属于Orange类别，但特征值Long独自就能使该类别的后验概率归零，从而导致分类决策错误。

朴素贝叶斯模型的缺点

$$P(x_i \mid C_k) = \frac{\text{count}(x_i, C_k)}{\text{count}(C_k)}$$

如何解决零概率问题?

- 拉普拉斯平滑（Laplace Smoothing）

给每个可能的特征值加一个小常数（通常是 1），避免零概率。

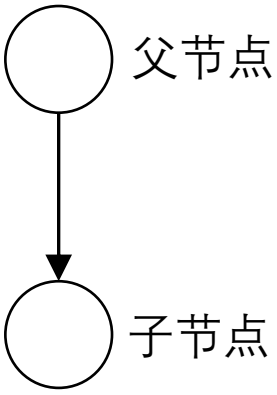
$$P(X_i = x_i | C_k) = \frac{\text{count}(X_i = x_i, C_k) + 1}{\text{count}(C_k) + V_i}$$

其中 V_i是特征 X_i的可能取值总数。这里**分母加** V_i是保证条件概率总和仍为 1，使其成为合法概率分布。

贝叶斯信念网络

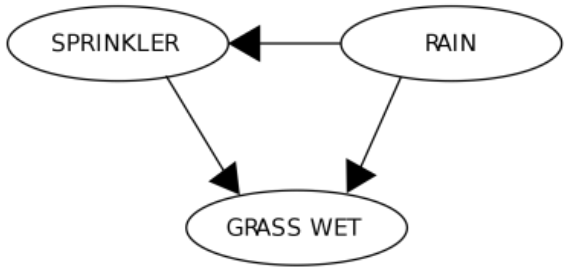
贝叶斯信念网络

- 贝叶斯网络（贝叶斯信念网络），是一种表示变量之间概率依赖关系的图模型（graph model）。它由以下两个主要组成部分构成：
- **网络结构 *G***：这是一个**有向无环图**（DAG），其中每个**节点（*V*）**代表一个随机变量。如果两个变量之间存在直接的概率依赖性，则它们之间会有一条有向边**（*E*）**相连。
- **参数*Θ***：参数是指每个节点（V）都具有的一个**条件概率表（CPT）**，它量化了在给定父节点的特定值的情况下，该节点取特定值的概率。



SPRINKLER

RAIN	T	F
F	0.4	0.6
T	0.01	0.99



RAIN

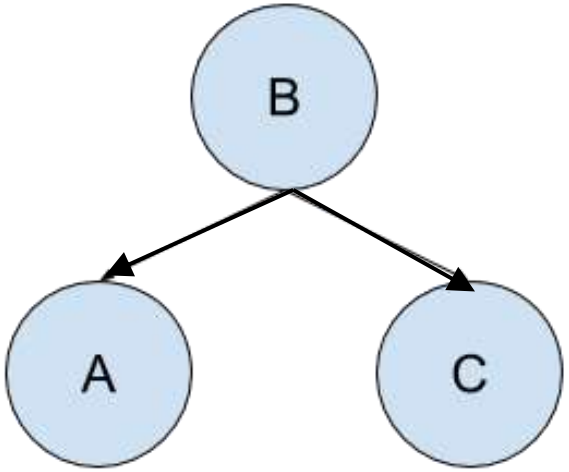
T	F
0.2	0.8

GRASS WET

SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

条件独立属性

- 贝叶斯网络的核心是揭示了变量间的**条件独立属性。**条件独立指的是在给定某些条件的情况下，多个随机变量相互独立。（贝叶斯网络通过图结构明确地表示哪些变量在给定其他变量（通常是父节点）时是条件独立的）。



这个图所代表的含义是：A, C在B发生的前提下，相互独立，即条件独立。

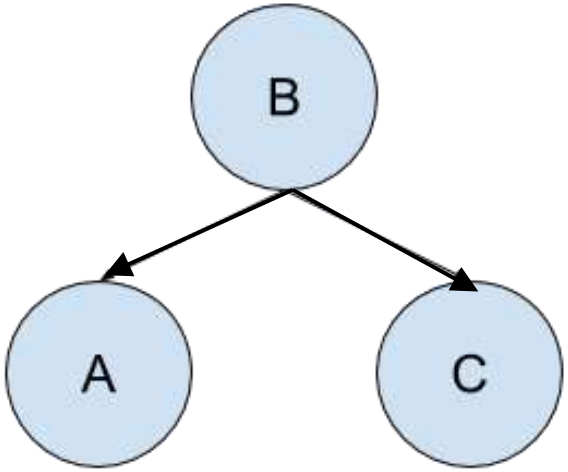
贝叶斯信念网络

- 贝叶斯网络核心应用是它能够**表示和处理多变量的联合概率分布**。它的应用具体包括以下几个方面:

  - **表示联合概率分布**
  - **条件概率查询和预测**
  - **推理和决策**

表示联合概率分布

A，B，C三个事件的联合概率分布是?



根据条件概率的定义:

$$P(A,B,C) = P(A,C \mid B)\,P(B)$$

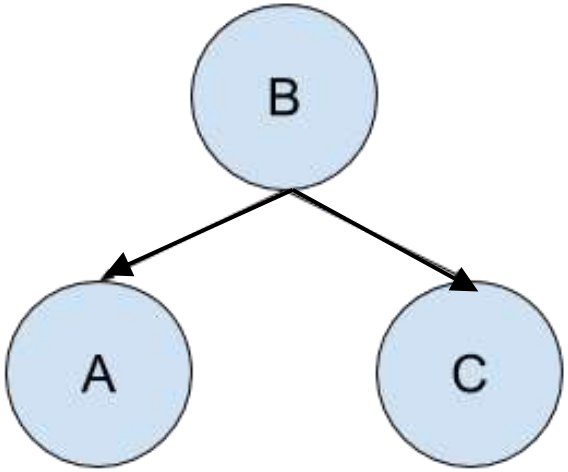
A, C在B发生的前提下，相互独立

$$P(A,C \mid B) = P(A \mid B)\,P(C \mid B)$$

$$P(A,B,C) = \underbrace{P(A \mid B)}_{\text{给定 } B \text{ 下 } A \text{ 的条件概率}} \times \underbrace{P(C \mid B)}_{\text{给定 } B \text{ 下 } C \text{ 的条件概率}} \times \underbrace{P(B)}_{\text{先验概率}}$$

表示联合概率分布

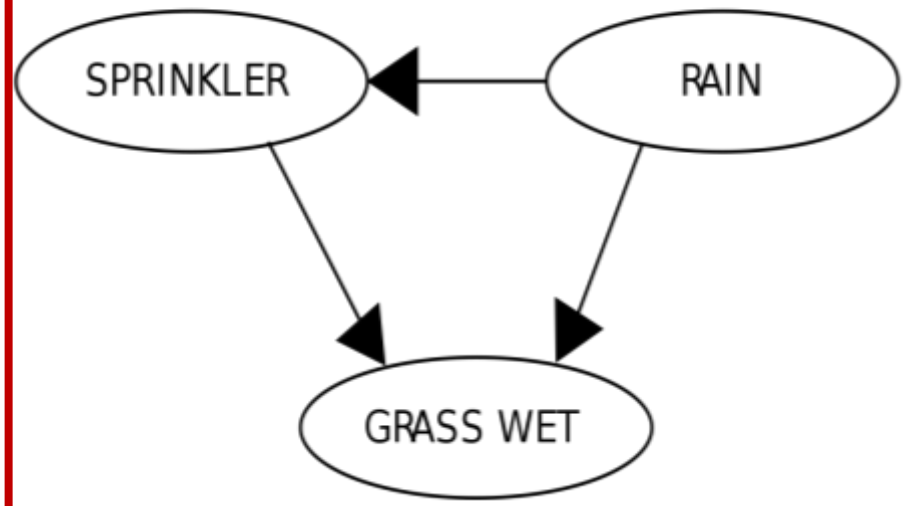
A，B，C三个事件的联合概率分布是?



- **B**没有父节点，所以它是一个先验概率P(B)
- **A**的父节点是B，因此A的条件概率是P(A|B)
- **C**的父节点是B：因此C的条件概率是P(C|B)

$$P(A,B,C) = \underbrace{P(A \mid B)}_{\text{给定 } B \text{ 下 } A \text{ 的条件概率}} \times \underbrace{P(C \mid B)}_{\text{给定 } B \text{ 下 } C \text{ 的条件概率}} \times \underbrace{P(B)}_{\text{先验概率}}$$

表示联合概率分布

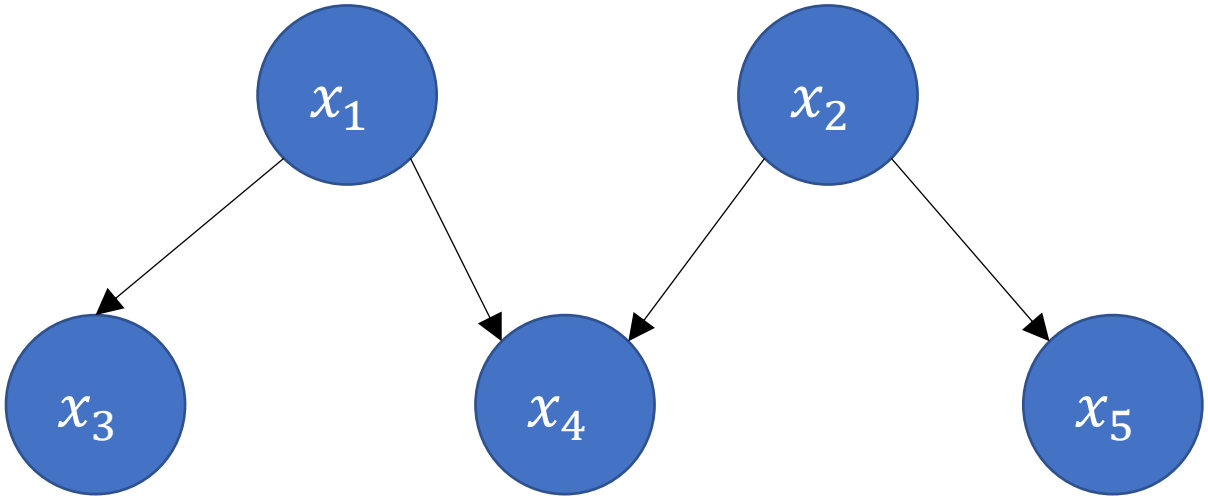


- **Rain** 没有父节点，所以它是一个先验概率P(Rain)。
- **Sprinkler** 的父节点是 Rain，因此Sprinkler的条件概率是P(Sprinkler|Rain)
- **GrassWet** 有两个父节点：Rain 和 Sprinkler。所以它的条件概率是P(GrassWet|Rain,Sprinkler)

将三个部分相乘，就得到了变量间的联合概率分布是:

$$\Pr(G, S, R) = \Pr(G \mid S, R) \Pr(S \mid R) \Pr(R)$$

表示联合概率分布

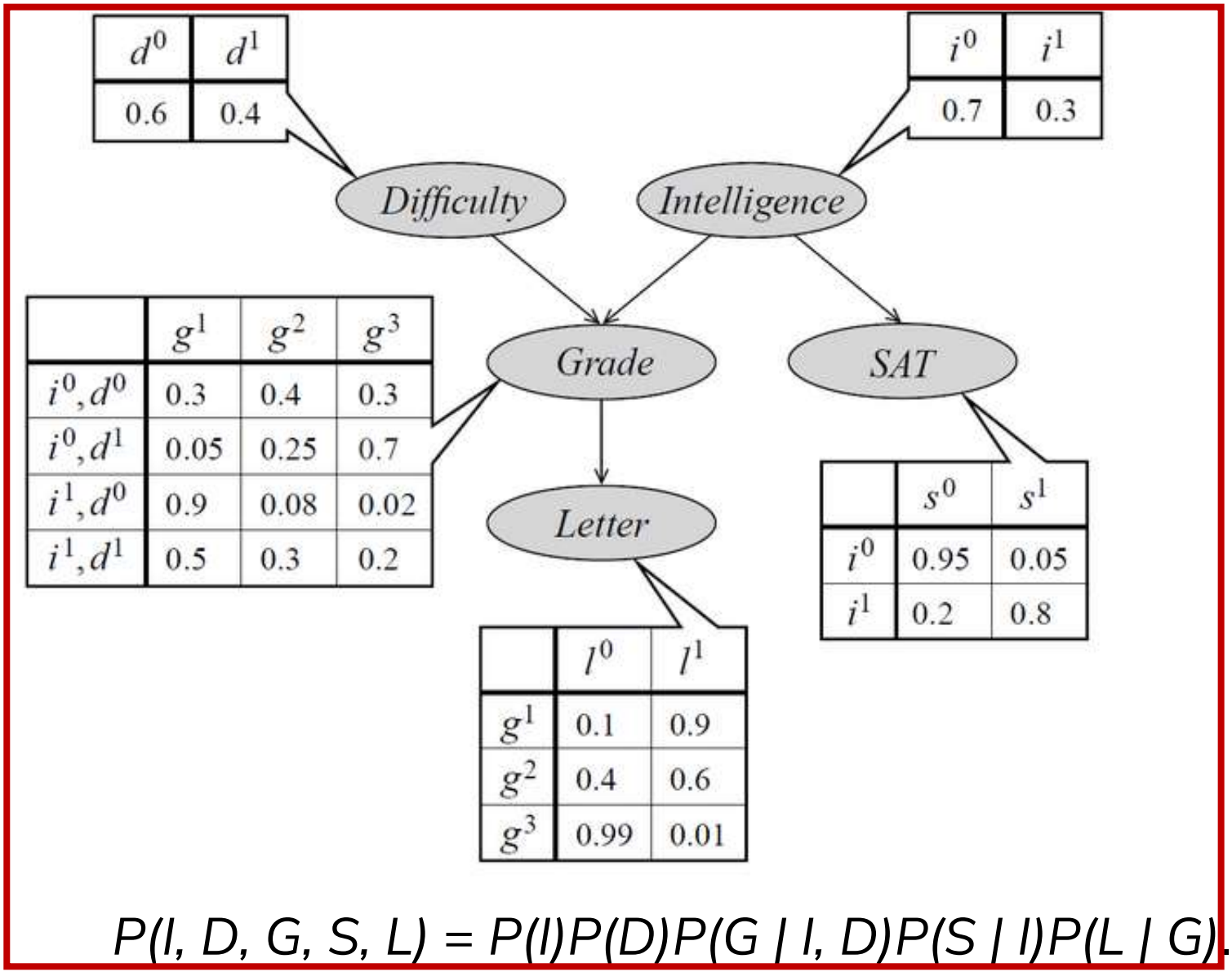


某个贝叶斯网络

它所表示的变量间的联合概率分布是:

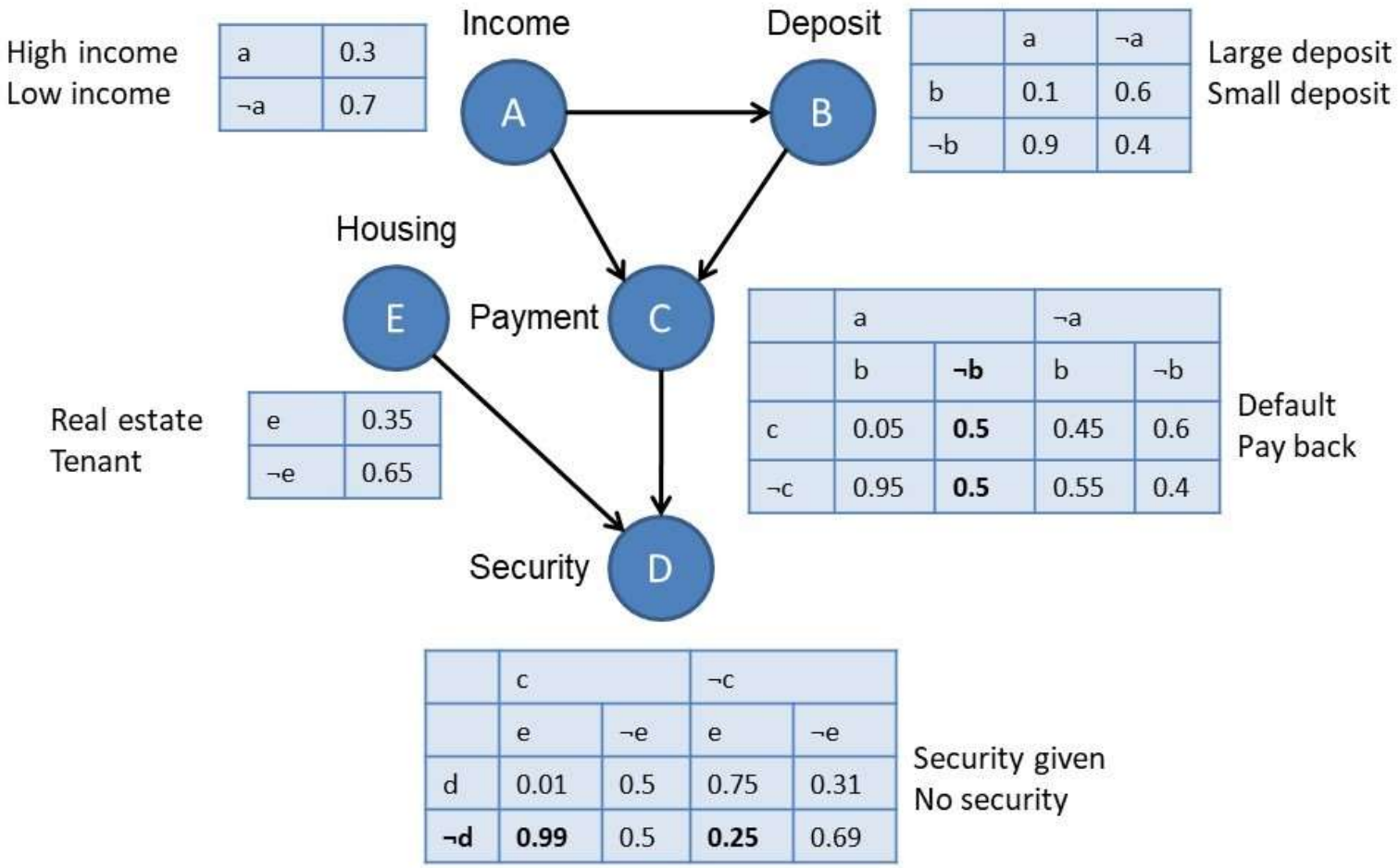
$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2)P(x_3 \mid x_1)P(x_4 \mid x_1, x_2)P(x_5 \mid x_2)$$

表示联合概率分布



$$P(I, D, G, S, L) = P(I)P(D)P(G \mid I, D)P(S \mid I)P(L \mid G).$$

表示联合概率分布



a	0.3
¬a	0.7

	a	¬a
b	0.1	0.6
¬b	0.9	0.4

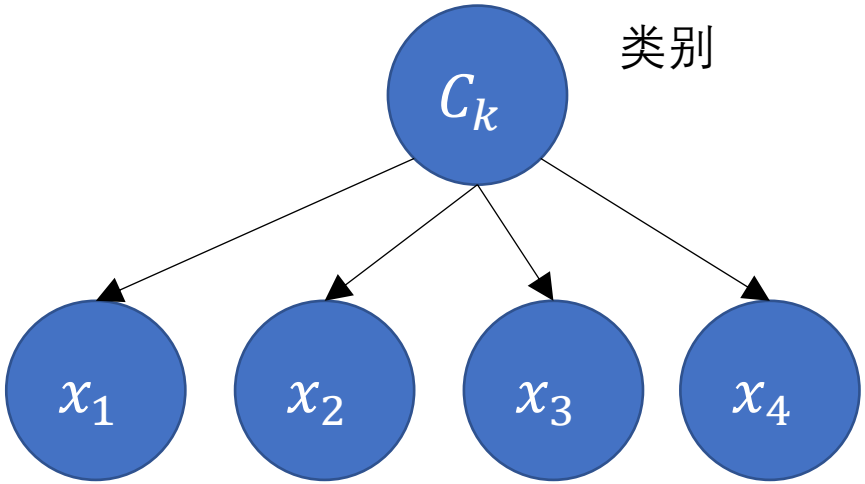
	a		¬a	
	b	**¬b**	b	¬b
c	0.05	**0.5**	0.45	0.6
¬c	0.95	**0.5**	0.55	0.4

	e	¬e
e	0.35	
¬e	0.65	

	c		¬c	
	e	¬e	e	¬e
d	0.01	0.5	0.75	0.31
¬d	**0.99**	0.5	**0.25**	0.69

表示联合概率分布

- 朴素贝叶斯是贝叶斯网络的一个特例
- 朴素贝叶斯如果用贝叶斯网络来表示



（给定类别下，特征相互独立）

通过贝叶斯网络推出朴素贝叶斯：

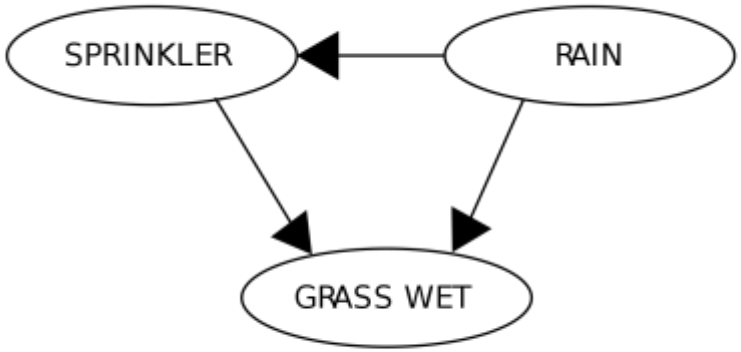
$$P(C_k|x_1,x_2,x_3,x_4)=\frac{P(C_k,x_1,x_2,x_3,x_4)}{P(x_1,x_2,x_3,x_4)}$$

$$=\frac{p(C_k)\times p(x_1|C_k)\times p(x_2|C_k)\times p(x_3|C_k)\times p(x_4|C_k)}{p(x_1,x_2,x_3,x_4)}$$

条件概率查询和预测

- 贝叶斯网络可以用来查询条件概率。例如，可以查询在给定某些证据变量的情况下其他变量的条件概率。

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



RAIN	
T	F
0.2	0.8

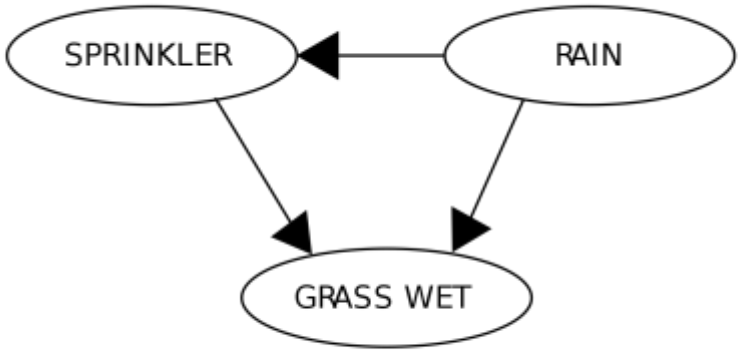
		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

问：浇水器开启，下雨同时发生的情况下，草湿的概率是多少？

条件概率查询和预测

- 贝叶斯网络可以用来查询条件概率。例如，可以查询在给定某些证据变量的情况下其他变量的条件概率。

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99



RAIN T	RAIN F
0.2	0.8

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

P(G=T|S=T, R=T) = 0.99

问：浇水器开启，下雨同时发生的情况下，草湿的概率是多少？

条件概率查询和预测

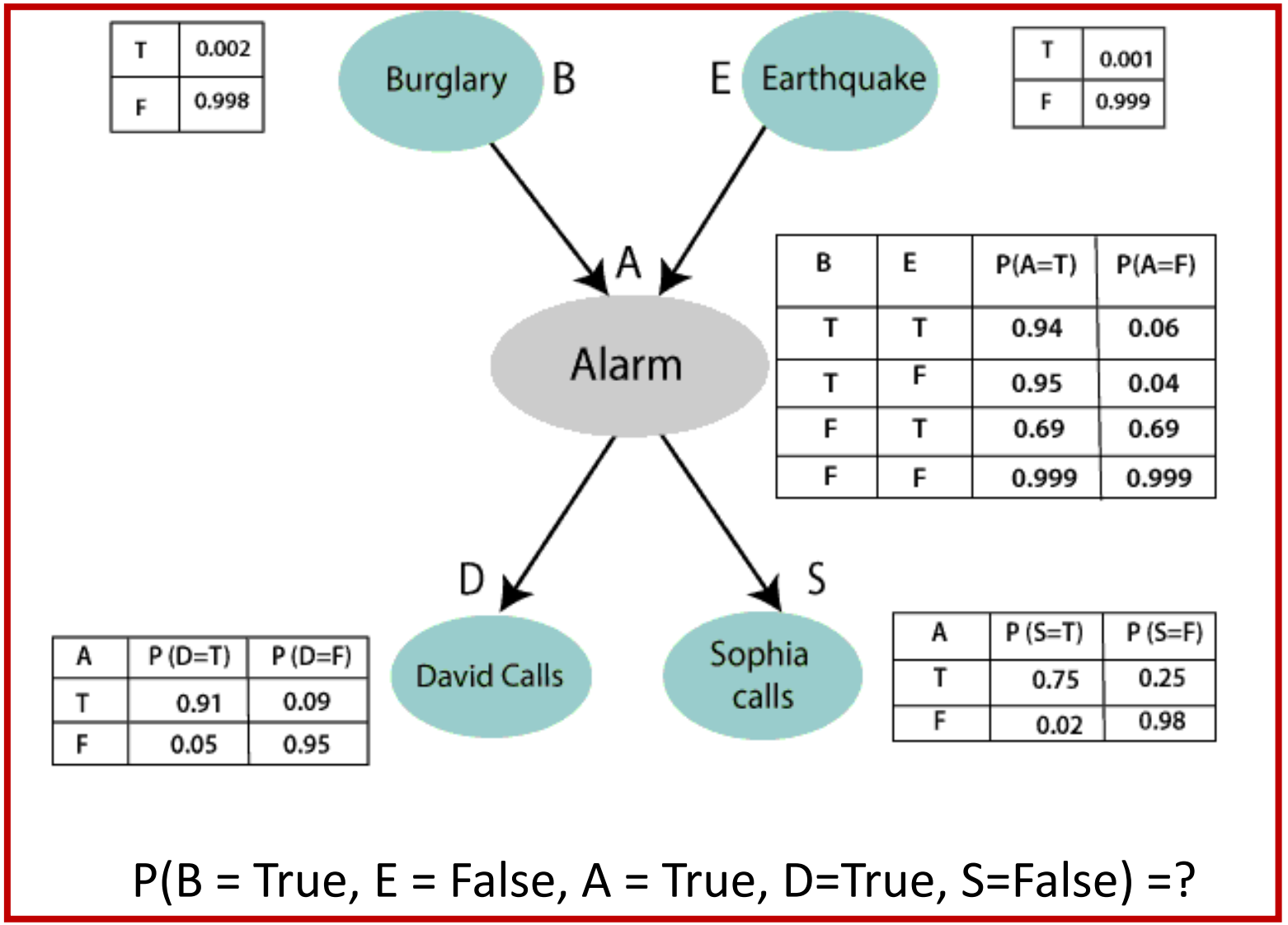
- 草湿，浇水器开启，下雨这三者同时发生的概率是?

$$\begin{aligned}\Pr(G=T,S=T,R=T) &= \Pr(G=T \mid S=T,R=T)\Pr(S=T \mid R=T)\Pr(R=T)\\ &= 0.99 \times 0.01 \times 0.2\\ &= 0.00198.\end{aligned}$$

- 如果草湿了，那么下雨的概率是多少?

$$\Pr(R=T \mid G=T) = \frac{\Pr(G=T,R=T)}{\Pr(G=T)} = \frac{\sum_{x\in\{T,F\}}\Pr(G=T,S=x,R=T)}{\sum_{x,y\in\{T,F\}}\Pr(G=T,S=x,R=y)}$$

条件概率查询和预测



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

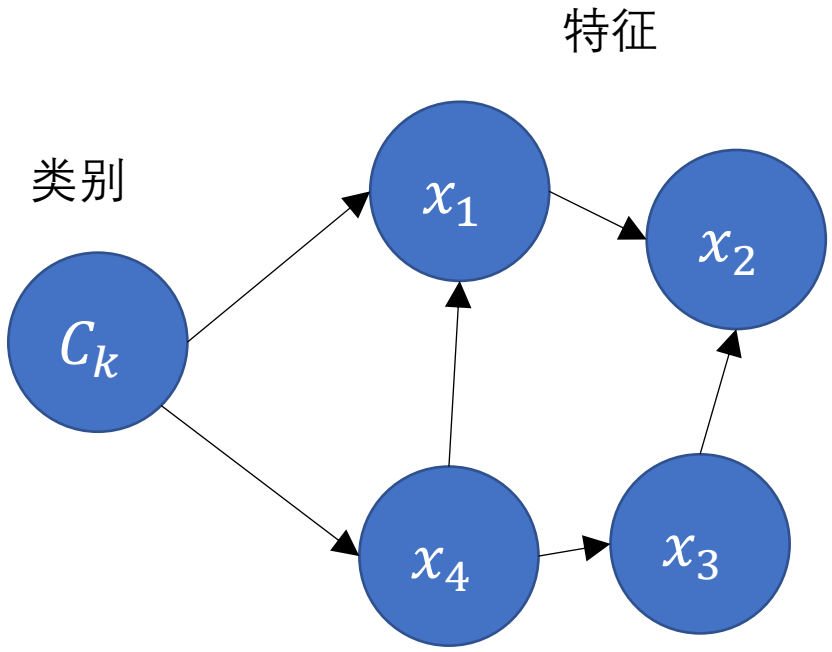
A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

P(B = True, E = False, A = True, D=True, S=False) =?

推理和决策

- 分类问题

将输入特征X和目标变量C_k 作为网络节点，建模节点间的依赖关系构建条件概率表，计算后验概率。

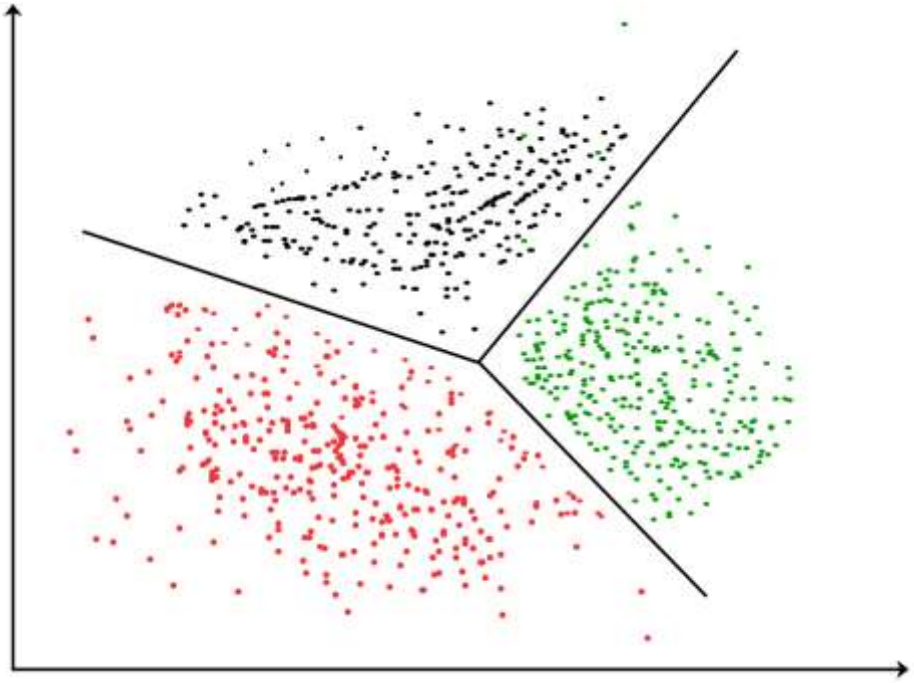


计算后验概率 $P(C_k|x_1,x_2,x_3,x_4)=\frac{P(C_k,x_1,x_2,x_3,x_4)}{P(x_1,x_2,x_3,x_4)}$

聚 类

聚类问题

- 无监督学习里典型例子就是**聚类**。
- 聚类是**将样本集合中相似的样本（实例）分配到相同的类，不相似的样本分配到不同的类。**与分类不同，聚类并不关心分的这一类是什么。
- 聚类的核心概念是**相似度**（similarity）或**距离**（distance）



- 聚类算法涉及到两个基本问题：**距离计算，性能度量**

聚类问题：距离计算

- 因为相似度直接影响聚类的结果，所以**距离和相似度选择是聚类的根本问题。**
- 有多种相似度或距离的定义（参见KNN一节），也可根据需要自己定义，但一般需满足如下条件：

非负性： $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geqslant 0$

同一性： $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$ 当且仅当 $\boldsymbol{x}_i = \boldsymbol{x}_j$

对称性： $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{dist}(\boldsymbol{x}_j, \boldsymbol{x}_i)$

直递性： $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leqslant \mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) + \mathrm{dist}(\boldsymbol{x}_k, \boldsymbol{x}_j)$

聚类问题：性能度量

性能度量的作用：

- 评估聚类效果的好坏
- 作为聚类的优化目标

足够“好的”聚类？

- 同一簇的样本尽可能彼此相似。（“簇内相似度”高）
- 不同簇的样本尽可能地不同。（“簇间相似度”低）

聚类问题：性能度量

轮廓系数（Silhouette Score）

轮廓系数综合考虑了聚类的**紧密度**和**分离度**，衡量了同一簇内样本之间的紧密程度的同时，又衡量了样本与其他簇之间的分离度。其值介于[-1, 1]之间，值越大表示聚类效果越好。

1. 对每个样本，计算其与同簇其他点的平均距离 $a(i)$（簇内紧密度）。
2. 计算该样本与最近其他簇中所有点的平均距离 $b(i)$（簇间分离度 ）。
3. 计算轮廓系数

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

对于聚类中的所有样本，我们可以计算所有样本的轮廓系数的平均值来得到**总体轮廓系数**：

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

其中，N 是样本总数，$s(i)$ 是样本 i的轮廓系数。

聚类问题：性能度量

轮廓系数（Silhouette Score）

- 如果聚类效果好，则$a(i)$很低，$b(i)$很高，那么

$$S(i) = \frac{b(i) - a(i)}{b(i)} = 1 - \frac{a(i)}{b(i)} \Longrightarrow$$ 接近于1

- 如果聚类效果差，则$a(i)$很高，$b(i)$很低，那么

$$S(i) = \frac{\mathrm{b}(i) - a(i)}{\mathrm{a}(i)} = \frac{b(i)}{a(i)} - 1 \Longrightarrow$$ 接近于-1

所以其值介于[-1, 1]之间，值越大表示聚类效果越好。

聚类问题：性能度量

Davies-Bouldin指数（DBI）

Davies-Bouldin 指数（DBI） 主要用于评估聚类结果中簇的**分离度**和**紧密度**。它能够反映簇之间的相似性和簇内的紧密度。

簇内距离s_i （簇的紧密度）
簇 C_i的紧密度 s_i通常定义为簇内所有样本到簇中心的平均距离。假设簇 C_i中有 $|C_i|$个样本，簇中心是这些样本的平均位置 μ_i，那么紧密度计算公式为：

$$s_i = \frac{1}{|C_i|}\sum_{x \in C_i} d(x, \mu_i)$$

其中，$d(x, \mu_i)$ 是样本 x到簇中心 μ_i的距离。

簇间距离 d_{ij}
簇 C_i和簇 C_j之间的距离通常是簇中心之间的距离，定义为：

$$d_{ij} = \| \mu_i - \mu_j \|$$

其中，μ_i 和 μ_j分别是簇 C_i和簇 C_j的中心，$\|\cdot\|$ 是欧几里得距离。

聚类问题：性能度量

Davies-Bouldin指数（DBI）

对于每一对簇 C_i和 C_j，可以 计算簇 i和簇 j之间的相似度 R_{ij}

簇内距离

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

簇间距离

最终，**Davies-Bouldin 指数**是所有簇对的最大相似度的平均值：

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

DBI 越小，簇之间的分离度较大且簇内的紧密度较强，聚类效果好。相反，DBI 越大，表示聚类效果较差，簇之间不够分离或簇内样本过于分散。

k-means算法

k-means算法（k均值聚类）

- k-means 是最常用的**基于欧式距离**的聚类算法。
- k-means 聚类是硬“聚类”，因为每个样本只能被分配到一个簇中。也就是说，样本只能属于一个簇，且簇之间是互不重叠的。
- **k-means 的目标是将 n 个样本分到 k 个不同的类或簇中**，其中 k是一个预设的超参数，通常满足 $k < n$。假设有 k个簇 $G_1, G_2, \ldots, G_k$，这些簇形成了对样本集合 X的划分，并且满足以下条件：

  簇之间互不重叠：

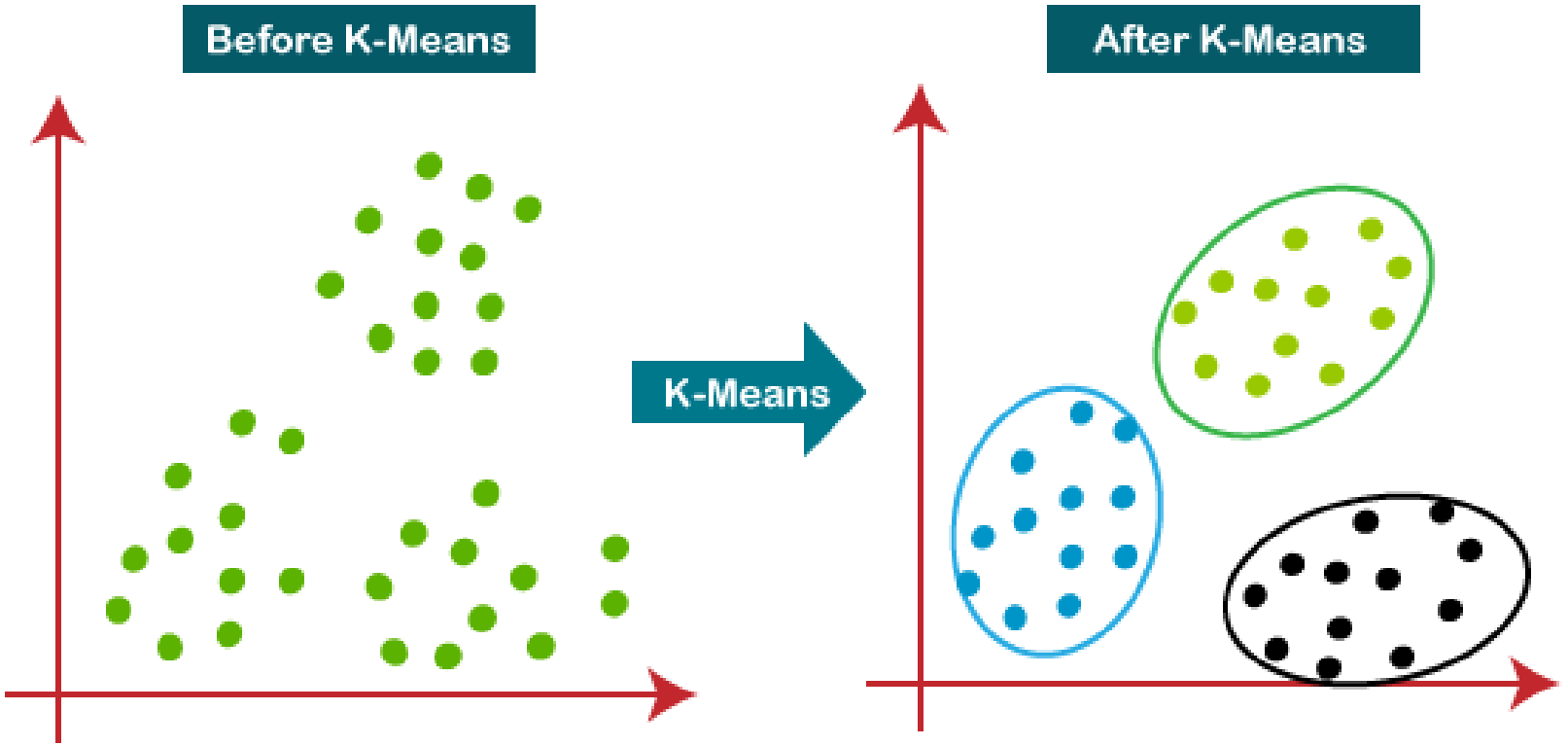
$$G_i \cap G_j = \emptyset \quad \text{for} \quad i \neq j$$

  所有簇的并集覆盖整个样本集 D：

$$\bigcup_{i=1}^{k} G_i = D$$

k-means算法（k均值聚类）

直观来看，K均值聚类做的是这样一件事：



k-means算法

$k-means$算法将欧氏距离平方作为样本间距离

策略

给定样本集$D=\{x_1,x_2,\dots,x_m\}$,“k均值”$(k-means)$算法针对聚类所得簇划分$C=\{C_1,C_2,\dots,C_k\}$最小化平方误差

$$E=\sum_{i=1}^{k}\sum_{x\in C_i}\|x-\mu_i\|_2^2$$

该式是K-means算法的目标函数

其中$\mu_{\mathrm{i}}=\frac{1}{|C_i|}\sum_{x\in C_i}x$是簇$C_i$的均值向量，也称为簇$C_i$ 的质心。该式也称为**聚类误差平方和（SSE）**或**簇内总平方和（WSS）**。

k-means 算法的策略就是最小化这个聚类误差平方和，使得同一个簇内的点彼此尽量相似（距离簇中心更近）。

k -means算法

算法步骤

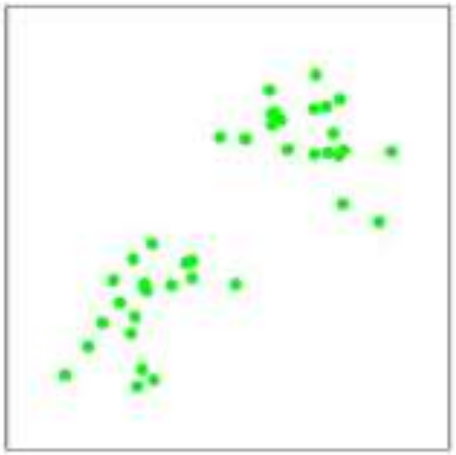
理解!

1. 选择初始化的 k 个样本作为初始聚类中心 $a = a_1, a_2, \ldots, a_k$;
2. 针对数据集中每个样本x_i，计算它到这 k个聚类中心的距离并将其分到距离最小的聚类中心所对应的类中;
3. 针对每个类别a_j，重新计算它的聚类中心（即属于该类的所有样本的质心）;
4. 重复上面 2 3 两步操作，直到达到某个中止条件（迭代次数 max_iter 、收敛阈值 tol，当簇中心的变化量小于该值时，算法停止迭代，通常默认为 1e-4，簇中心的变化小于这个值时停止。）。

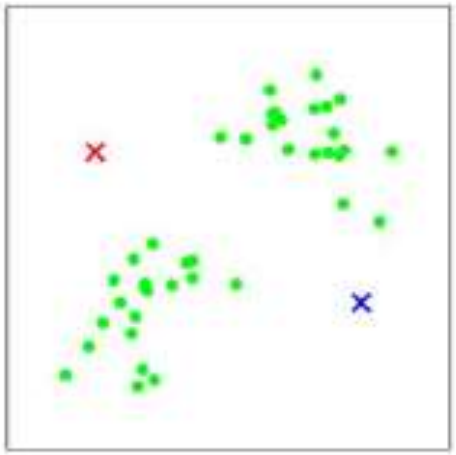
k-means算法

算法例

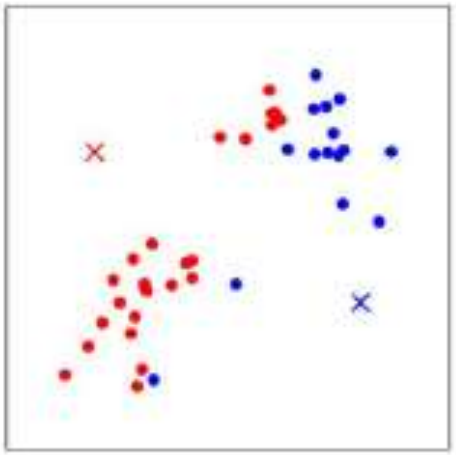
- 假设k=2。



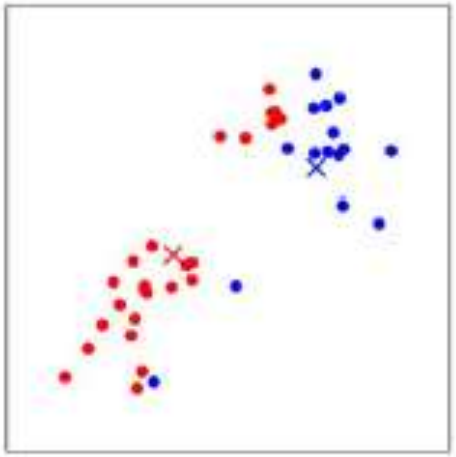
(a)



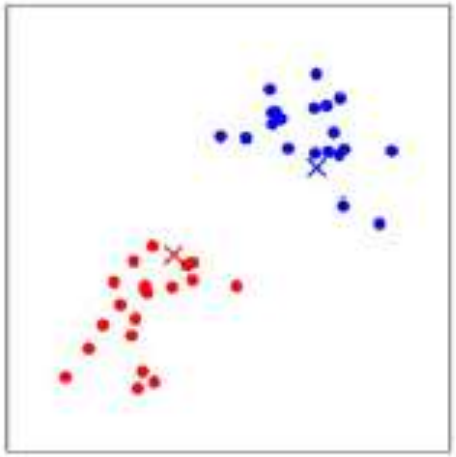
(b)



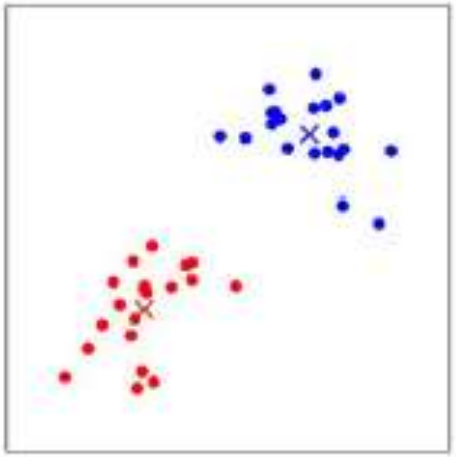
(c)



(d)



(e)



(f)

k-means算法-例

给定含有5个样本的集合

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

试用k均值聚类算法将样本聚到2个类中

质心	距离（欧式距离平方）				
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (0,2)$	0	4	5	29	25
$x_{c2} = (0,0)$	4	0	1	25	29

$$\Rightarrow \quad \begin{matrix} G_1 = \{x_1, x_5\} \\ G_2 = \{x_2, x_3, x_4\} \end{matrix} \quad \Rightarrow \quad \begin{matrix} x_{c1} = (2.5,2) \\ x_{c2} = (2,0) \end{matrix}$$

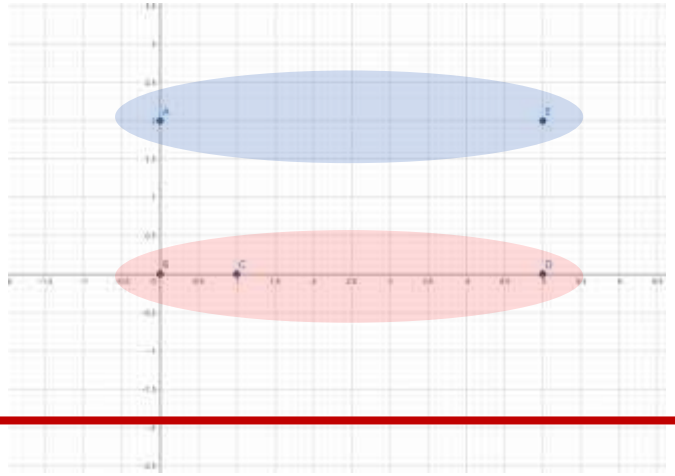
k-means算法-例

质心	距离（欧式距离平方）				
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (2.5,2)$	6.25	10.25	4.25	10.25	6.25
$x_{c2} = (2,0)$	8	4	1	9	13

$$\Rightarrow \quad \begin{matrix} G_1 = \{x_1, x_5\} \\ G_2 = \{x_2, x_3, x_4\} \end{matrix} \quad \Rightarrow \quad \begin{matrix} x_{c1} = (2.5,2) \\ x_{c2} = (2,0) \end{matrix}$$

由于得到的新的类没有改变，聚类停止。得到最终的聚类结果：

$$\begin{matrix} G_1 = \{x_1, x_5\} \\ G_2 = \{x_2, x_3, x_4\} \end{matrix}$$



k-means算法-例

给定含有5个样本的集合

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

试用k均值聚类算法将样本聚到2个类中

质心	距离（欧式距离平方）				
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (0,2)$					
$x_{c2} = (5,2)$					

k-means算法-例

第一次迭代

质心	距离（欧式距离平方）				
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (0,2)$	0	4	5	29	25
$x_{c2} = (5,2)$	25	29	20	4	0

$\Rightarrow$ $G_1 = \{x_1, x_2, x_3\}$, $G_2 = \{x_4, x_5\}$ $\Rightarrow$ $x_{c1} = (0.33, 0.67)$, $x_{c2} = (5,1)$

第二次迭代

质心	距离（欧式距离平方）				
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (0.33,\ 0.67)$	0.19	0.56	0.89	22.22	23.56
$x_{c2} = (5,1)$	26	26	17	1	1

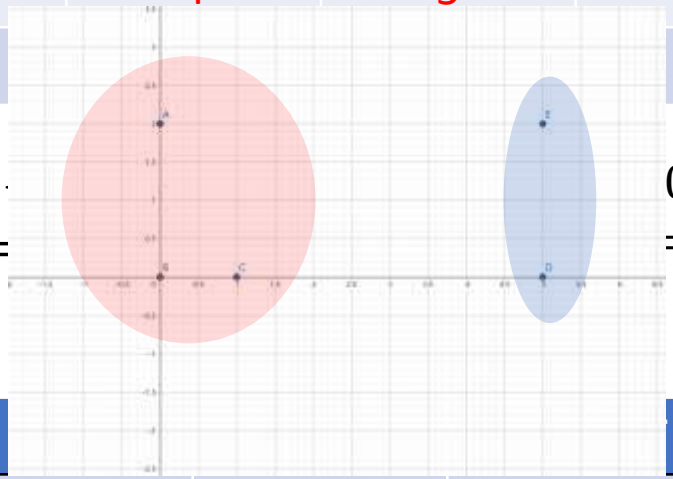
$\Rightarrow$ $G_1 = \{x_1, x_2, x_3\}$, $G_2 = \{x_4, x_5\}$

同样一组数据，因为初始点的不同，导致聚类结果不一样

k-means算法-例

第一次迭代

质心	距离（欧式距离平方）				
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (0,2)$	0	4	5	29	25
$x_{c2} = (5,2)$	25			4	0



$G_1 =$ 0.33,0.67)

$G_2 =$ $= (5,1)$

第二次迭代

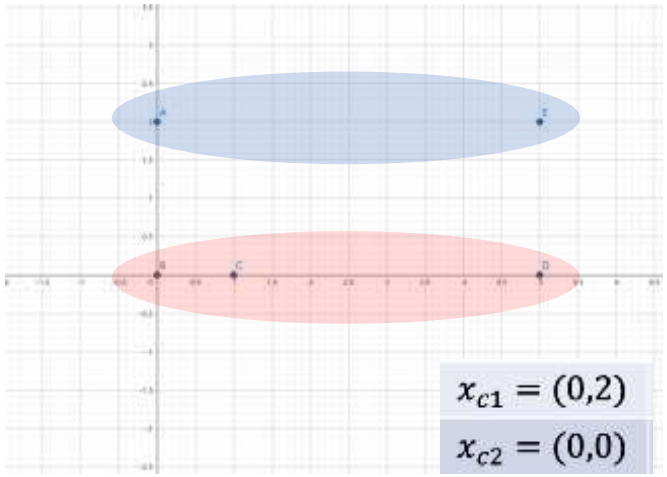
质心		平方）			
	$x_1 = (0,2)$	$x_2 = (0,0)$	$x_3 = (1,0)$	$x_4 = (5,0)$	$x_5 = (5,2)$
$x_{c1} = (0.33,\ 0.67)$	0.19	0.56	0.89	22.22	23.56
$x_{c2} = (5,1)$	26	26	17	1	1

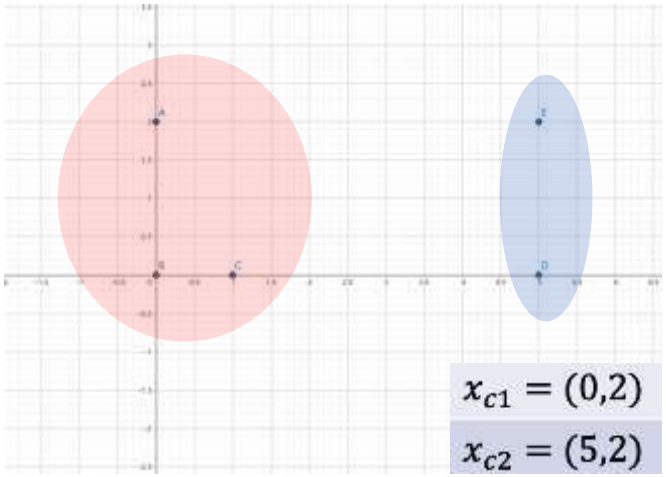
$$G_1 = \{x_1, x_2, x_3\}$$
$$G_2 = \{x_4, x_5\}$$

同样一组数据，因为初始点的不同，导致聚类结果不一样

初始质心的选择

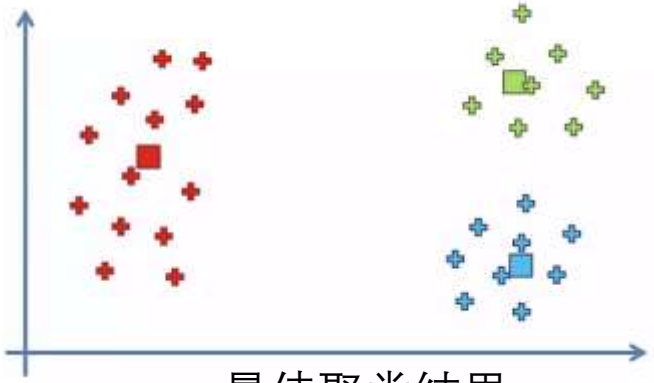
在 *K-means* 算法中，初始化簇中心（或称质心）是影响聚类效果和收敛速度的关键因素。不同的初始质心可能导致收敛到不同的局部最优解，从而影响聚类效果。





同样的一组数据，因为初始中心选择的不同，得到不同的聚类结果

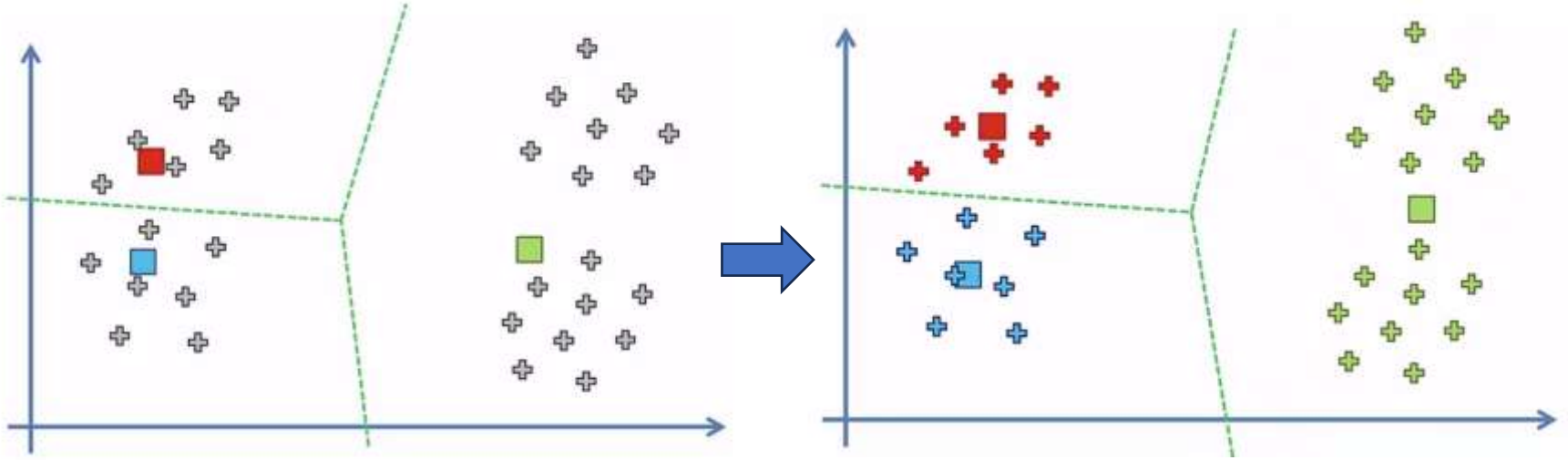
初始质心的选择



最佳聚类结果

随机初始化（Random Initialization）

- 随机选择 k个样本点作为簇中心。
- 簇中心是完全随机地选择的，**没有任何约束**。因此，可能会有两个或多个簇中心在数据空间中非常接近。这会导致某些簇在初始阶段几乎没有数据点被分配，从而使算法容易陷入**局部最优解**，并且收敛速度较慢。



初始质心的位置

聚类后的结果

初始质心的选择

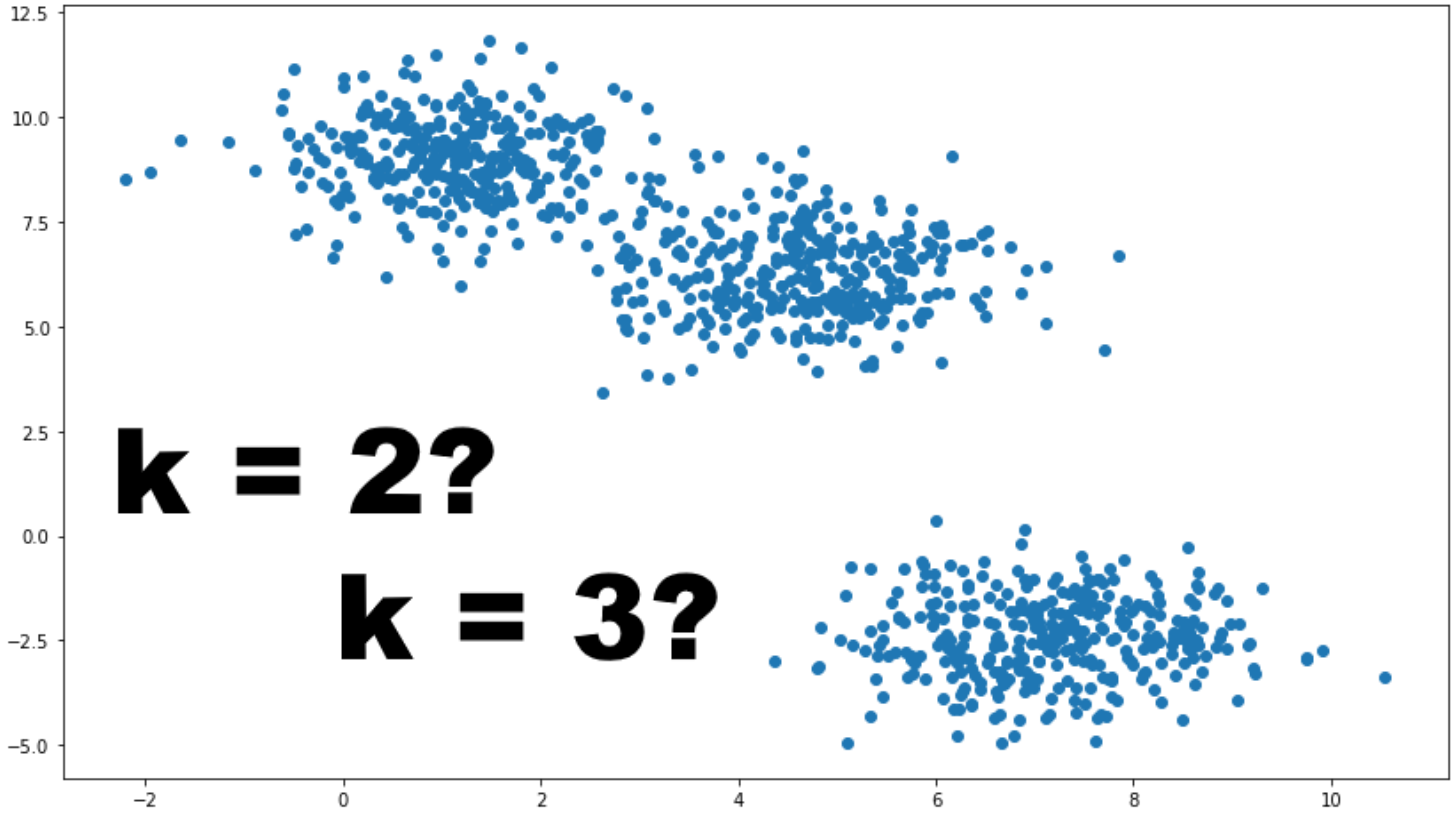
K-means++

1. 选择第一个质心：随机选择一个数据点作为第一个质心。
2. 选择后续质心：对于每个未被选择的点，计算该点到已经选择的质心的最小距离，并选择其中最大的那个点作为下一个质心。
3. 重复：继续这个过程，直到选择出 k 个质心。

确保每个新选择的质心都尽可能远离已经选择的质心，从而使得最终的质心分布更加均匀，减少局部最优的可能性，提升聚类效果。

k值的选择

- K的取值需要事先确定，然而在无监督聚类任务上，由于并不知道数据集究竟有多少类别，所以很难确定合适的K的取值。



k值的选择

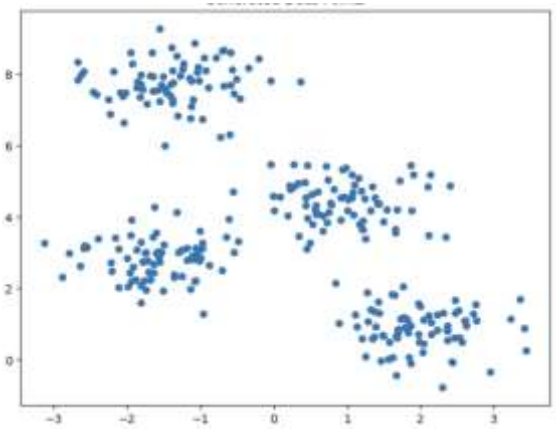
肘部法：观察“拐点”

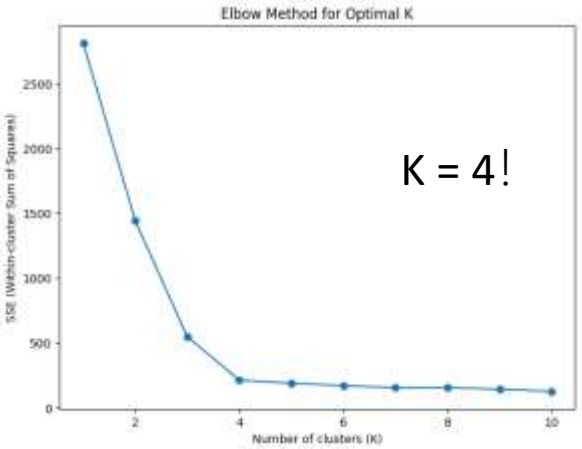
聚类误差平方和（SSE）

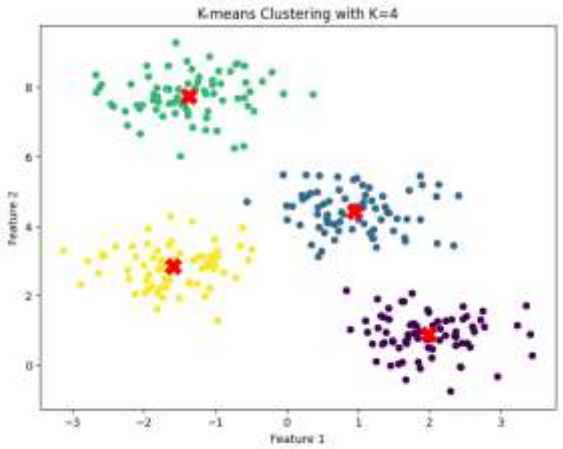
即K-means算法的优化目标，只衡量簇内紧密度。

通过计算不同 K 值下的**聚类误差平方和（SSE）或簇内总平方和（WSS）**，观察误差随 K 值变化的趋势。

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2$$



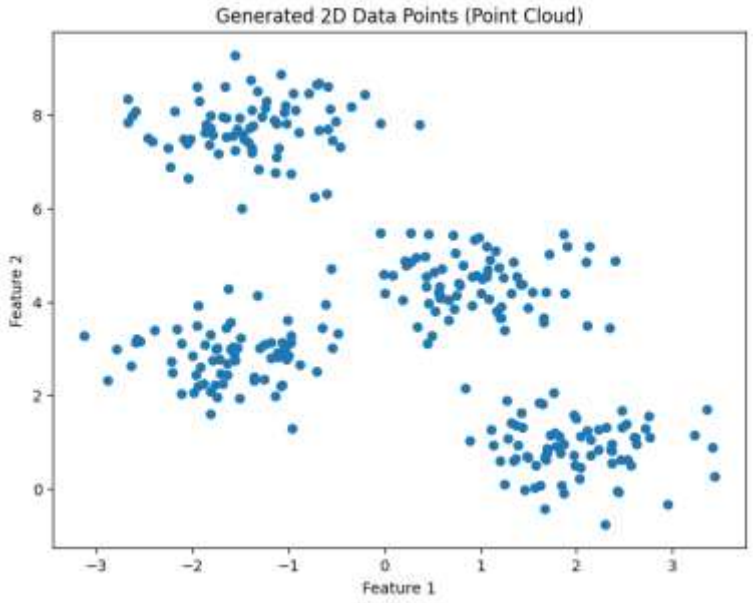


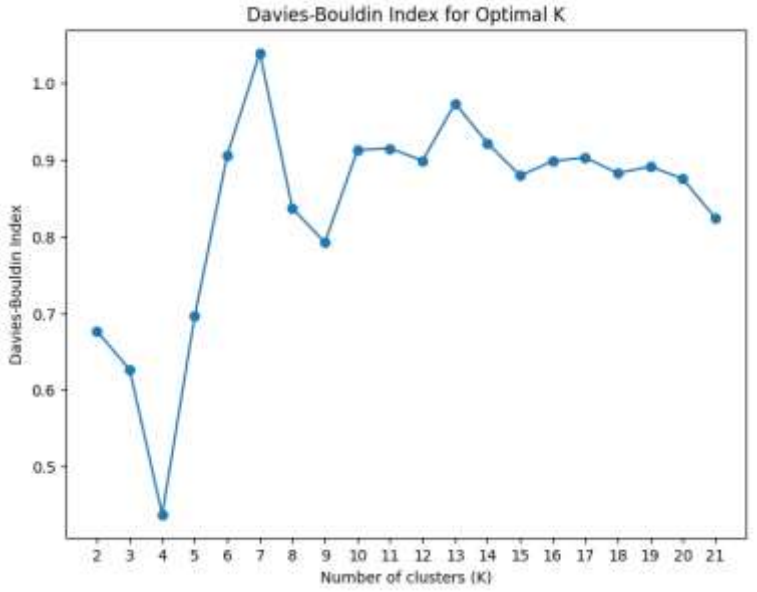


k值的选择

DB 指数

Davies-Bouldin 指数衡量簇内紧密度和簇间分离度。它计算的是每一对簇之间的相似度，值越小越好。

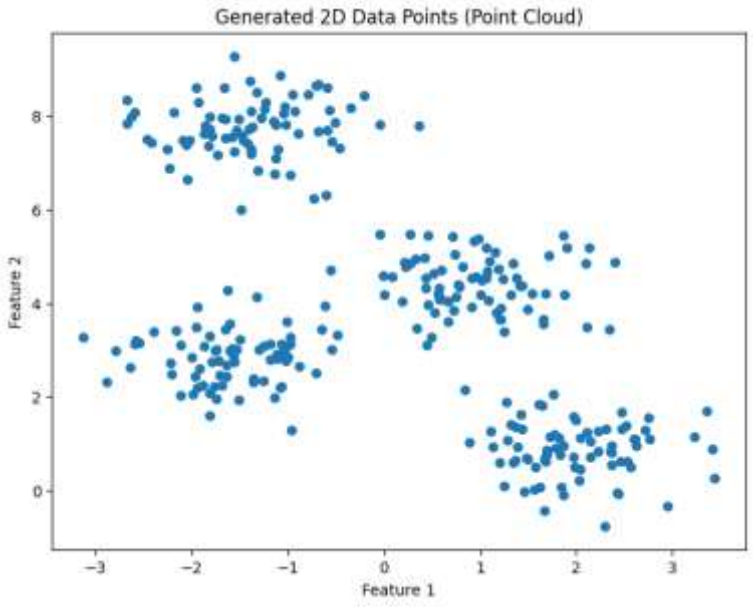


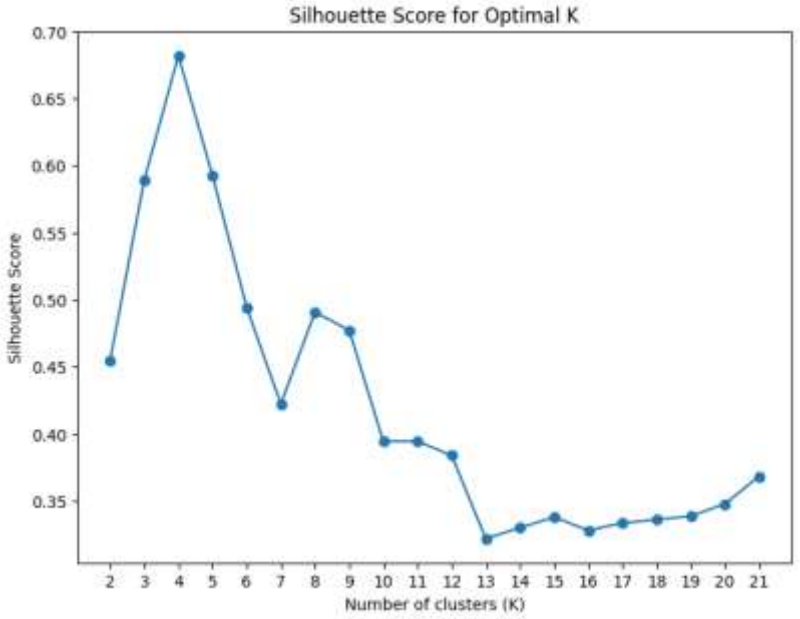


k值的选择

轮廓系数

轮廓系数衡量簇内紧密度和簇间分离度。其值范围在 -1 到 1 之间，值越高，聚类效果越好。

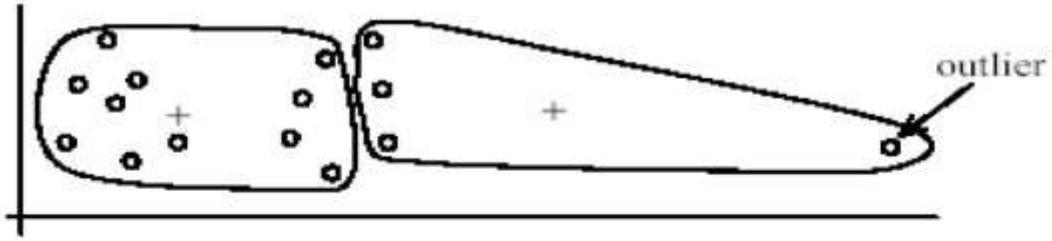




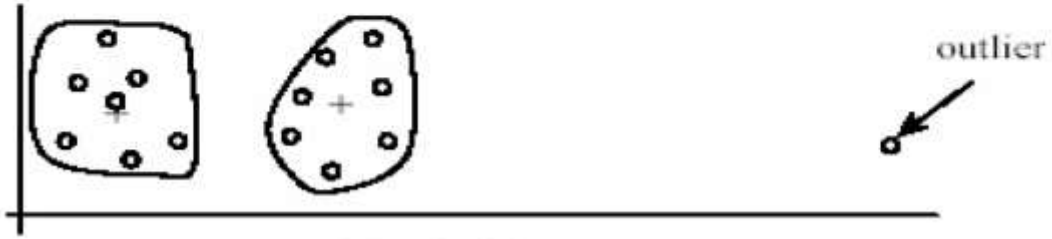
k-means算法特性

缺点：

1. **要确定K的值**：K-Means需要事先确定聚类数目K，这在无监督聚类任务中可能会很困难，因为通常无法事先知道数据集应该被分为多少个类别。
2. **对异常点敏感**：K-Means很容易受到异常点（离群值）的影响，因为它使用簇内样本均值来更新质心，异常点可能会导致质心偏移。



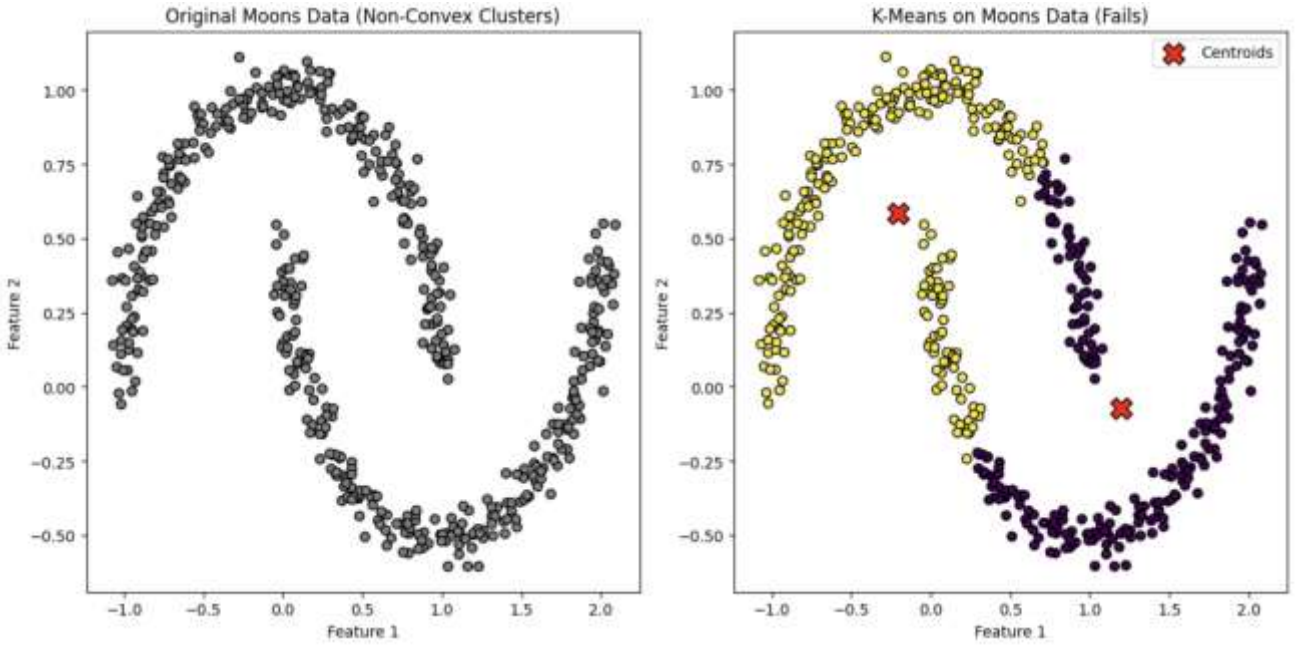
(A): Undesirable clusters



(B): Ideal clusters

k-means算法特性

3. **不适合非球形数据集**：K-Means 算法之所以更适合“球形”分布的簇，主要是因为它的聚类过程和使用的度量方式（欧几里得距离）使得每个簇的边界都呈现“离中心最近”的形状，这往往对应**近似球形**的区域。这种天然倾向于得到球形或相对均匀的簇的特点，使得K-means对于明显的非球形的数据集表现不佳。



k-means算法特性

优点：

1. **简单易实现**
   算法原理直观、流程清晰，便于编程实现和理解。
2. **计算速度快**
   由于每次迭代只涉及数据点与聚类中心的距离计算，算法在处理大规模数据集时能快速收敛。
3. **适用于大规模数据**
   k-means 的计算复杂度较低（通常为线性增长），因此能有效处理大数据集，具有良好的扩展性。

降维

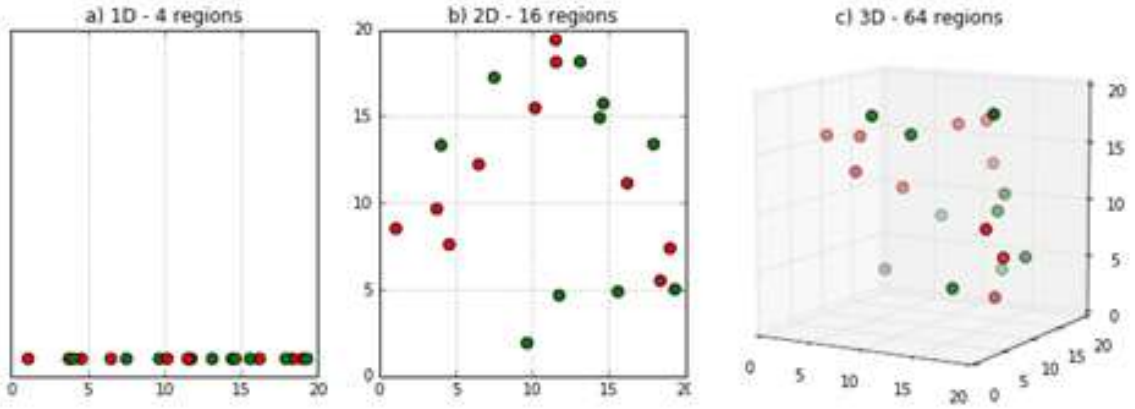
降维

- 降维是把原本在**高维空间**（很多特征、很多变量）中的数据，**转换成维度更低但仍能保留主要信息**的新表示。

为什么需要降维？

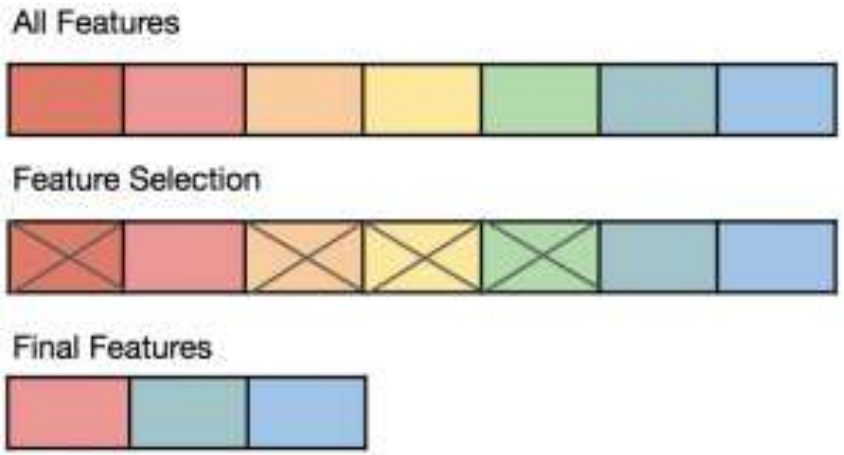
1. 去除冗余特征：有些特征彼此高度相关。
2. **减少噪声影响**：去掉不重要的维度，让模型更稳定。
3. 加快计算速度：维度少了，计算更快。
4. **便于可视化**：比如把高维数据（100维）投影到 2D/3D，方便画图观察。
5. **避免“维度灾难”**：高维空间中数据稀疏、距离失真，模型难以学习。

“维度灾难” ➡

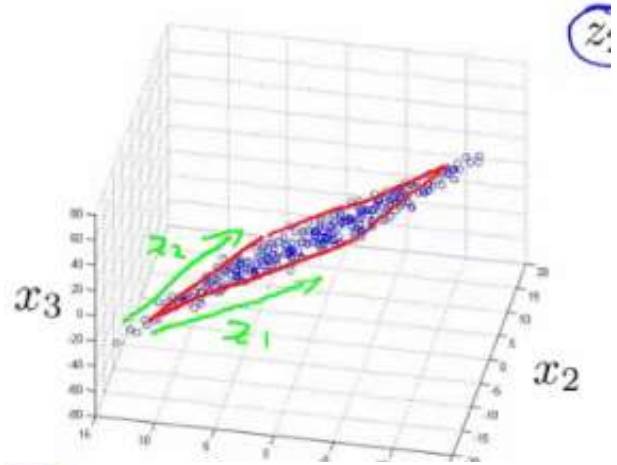


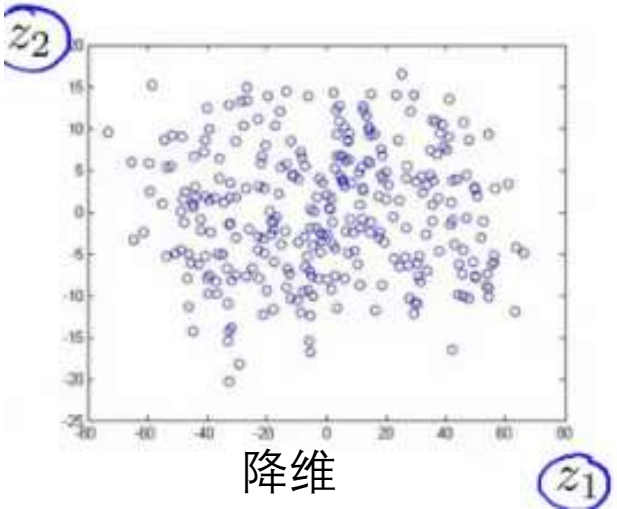
降维和特征选择

虽然都是为了**降低特征维度、简化模型、减少噪声、避免过拟合，也**都会让输入特征数量变少，但是他们有本质的不同!



特征选择





降维

常见的降维方法

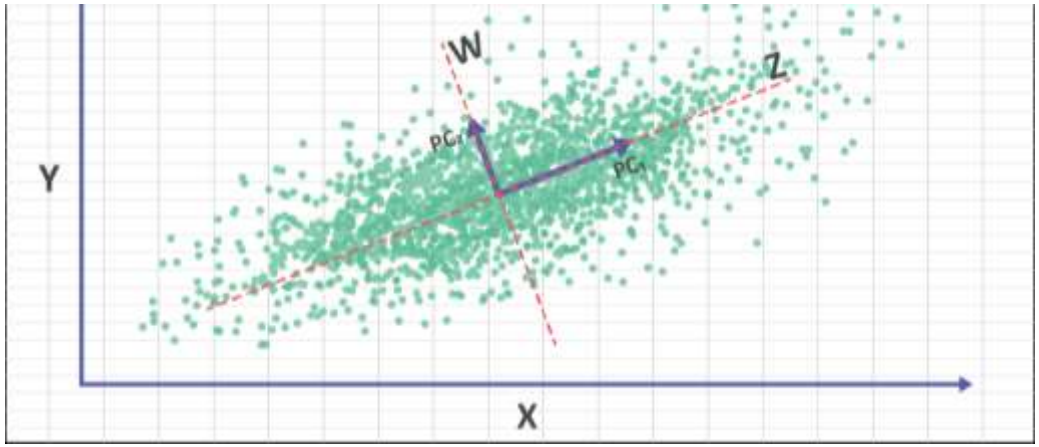
方法	类型	核心思想
PCA（主成分分析）	线性	找出方差最大的方向作为新坐标轴
t-SNE	非线性	保留局部相似性关系
UMAP	非线性	保留拓扑结构关系
Autoencoder（自编码器）	神经网络	通过网络学习压缩再重构
LDA（线性判别分析）	监督型	保留类别区分信息

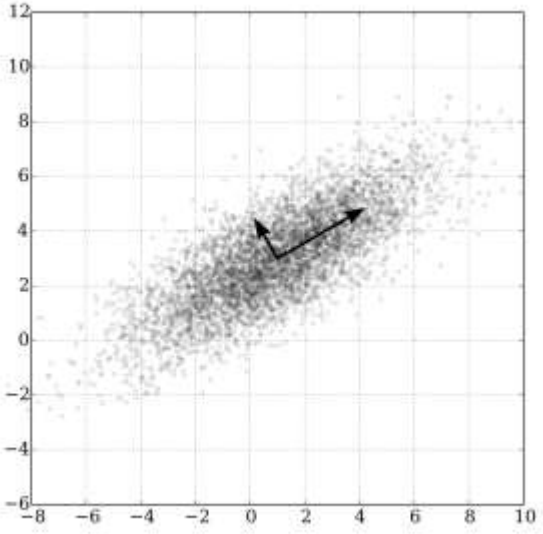
常见的降维方法

方法	类型	核心思想
PCA（主成分分析）	线性	找出方差最大的方向作为新坐标轴
t-SNE	非线性	保留局部相似性关系
UMAP	非线性	保留拓扑结构关系
Autoencoder（自编码器）	神经网络	通过网络学习压缩再重构
LDA（线性判别分析）	监督型	保留类别区分信息

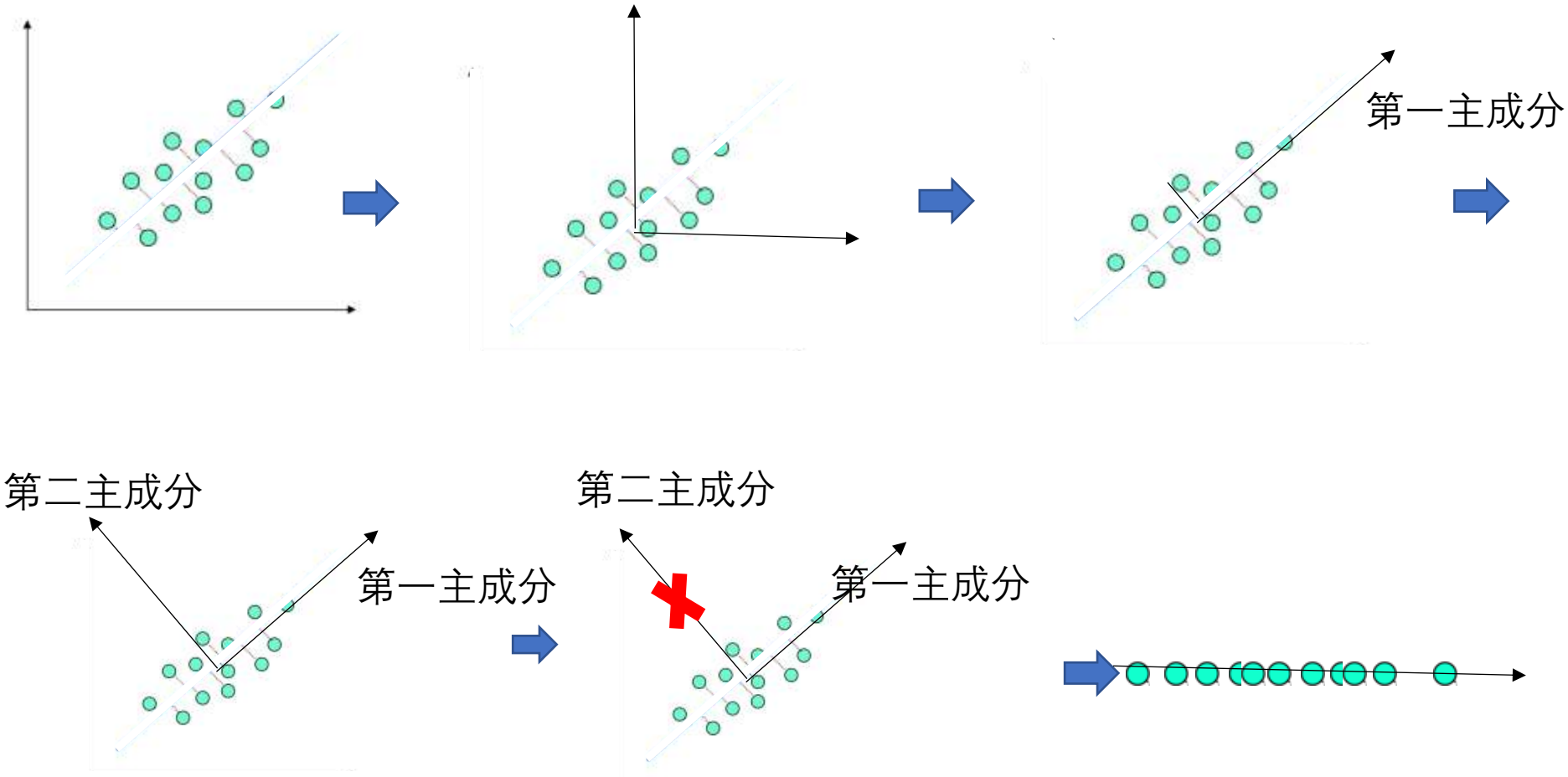
线性降维（PCA）

- 线性降维是通过对原始特征空间的坐标系进行**线性变换**，将数据映射到一个低维子空间中，从而减少特征的数量而保留尽可能多的信息。
- PCA是线性降维的代表性方法之一，它的优化目标是**找到一组新的基底（主成分），使得数据在这些基底上的投影方差最大**。
- 第一主成分，第二主成分，。。。





PCA的基本思路



PCA的主要步骤

1. 数据中心化

首先，需要将原始数据进行**中心化**。假设我们有一个 $m \times n$的数据矩阵 X，其中 m是样本数量，n 是特征数量。

每个特征的均值 μ_j可以通过以下公式计算：

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} X_{ij} \ \textit{(对每列求均值)}$$

然后，对每个特征减去其均值，使得每一列的均值为零：

$$X' = X - \mu$$

其中，X' 是中心化后的数据矩阵。

PCA的主要步骤

2. 计算协方差矩阵

协方差矩阵 C描述了不同特征之间的关系。它可以通过以下公式计算:

$$C = \frac{1}{m-1} X'^T X'$$

这里，X'^T 是 X'的转置，X' 是中心化后的数据矩阵。协方差矩阵是一个 $n \times n$的矩阵，表示数据中各特征之间的协方差。

3. 计算协方差矩阵的特征值和特征向量

对协方差矩阵 C进行特征值分解，我们可以得到特征值 λ和特征向量 v:

$$Cv = \lambda v$$

这里，v 是特征向量，表示数据中主要变化的方向；λ 是特征值，表示特征向量方向上的方差大小。特征值越大，代表数据在这个方向上的变化越大。

PCA的主要步骤

4. 选择主成分

PCA的目标是选择数据中方差最大的方向作为主成分。因此，我们会选择最大的几个特征值对应的特征向量。这些特征向量就代表了数据变化的**主要方向**。

假设特征值按从大到小排序，选择前 k个特征值对应的特征向量，组成矩阵 V_k:

$$V_k = [v_1, v_2, \ldots, v_k]$$

这些 k个特征向量就是我们选择的**主成分**，它们定义了新的数据空间。

5. 数据投影

最后，我们将原始数据投影到这些主成分上，得到降维后的数据。假设 X'是中心化后的数据，V_k 是选择的主成分矩阵，则投影后的数据 Y为:

$$Y = X'V_k$$

这里，Y 是降维后的数据矩阵，维度由 n降到 k（通常 $k < n$）。

PCA例

请利用PCA算法计算该数据集的主成分，并将数据降到1维

样本序号	特征1	特征2
1	1	-1
2	1	1
3	2	1
4	2	2
5	4	2

PCA例

Step 1：构建数据矩阵

$$X = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 2 & 1 \\ 2 & 2 \\ 4 & 2 \end{bmatrix}$$

Step 2：中心化数据

计算每列均值：

$$\mu_1 = \frac{1+1+2+2+4}{5} = \frac{10}{5} = 2$$

$$\mu_2 = \frac{-1+1+1+2+2}{5} = \frac{5}{5} = 1$$

PCA例

中心化后的数据矩阵 X':

$$X' = X - \mu = \begin{bmatrix} 1-2 & -1-1 \\ 1-2 & 1-1 \\ 2-2 & 1-1 \\ 2-2 & 2-1 \\ 4-2 & 2-1 \end{bmatrix} = \begin{bmatrix} -1 & -2 \\ -1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 2 & 1 \end{bmatrix}$$

Step 3：计算协方差矩阵

$$C = \frac{1}{m-1} X'^T X'$$

$$X'^T = \begin{bmatrix} -1 & -1 & 0 & 0 & 2 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\Rightarrow \quad C = \begin{bmatrix} 1.5 & 1 \\ 1 & 1.5 \end{bmatrix}$$

PCA例

Step 4：求特征值和特征向量

特征方程：

$$\det(C - \lambda I) = 0$$

$$\begin{vmatrix} 1.5 - \lambda & 1 \\ 1 & 1.5 - \lambda \end{vmatrix} = (1.5 - \lambda)^2 - 1 = 0$$

$$(1.5 - \lambda)^2 = 1 \implies 1.5 - \lambda = \pm 1$$

$$\lambda_1 = 1.5 + 1 = 2.5, \quad \lambda_2 = 1.5 - 1 = 0.5$$

对应的特征向量分别是：

$$v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

第一主成分　　第二主成分

PCA例

Step 5：选择主成分并降维

我们选择特征值 $\lambda_1 = 2.5$对应的特征向量作为投影基底，将将数据投影到1维空间。

$$v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

投影公式：

$$Y = X' v_1$$

最终，投影后的数据为：

$$Y = \begin{bmatrix} \frac{-3}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ \frac{3}{\sqrt{2}} \end{bmatrix}$$

PCA应用例：人脸识别

在**人脸识别**问题中，面临着数据维度过高的问题。每张人脸图像可以有成千上万的像素点（例如，一张640x480像素的图像有超过30万个像素）。为了从这些图像中提取有用的信息并用于识别，需要一种有效的降维方法。

Datasets

PCA

Eigenfaces

PCA应用例：人脸识别

- 每张图像都被投影到这个低维空间，得到一个表示该图像的低维特征向量。
- 在实际识别时，将待识别人脸图像投影到相同的Eigenfaces空间中，然后与已知身份的特征向量进行匹配，确定最相似的图像，完成身份识别。