

计算机模式识别与机器学习

—— 决策树



主讲：图像处理与模式识别研究所
赵群飞

邮 箱： zhaoqf@sjtu.edu.cn

办公室： 电院 2-441

电 话： 13918191860

第10章 决策树

- 本章学习目标
 - ✓ 理解决策树原理
 - ✓ 能够熟练运用迭代二叉树ID3算法
 - ✓ 掌握C4.5决策树算法
 - ✓ 了解CART算法

本章目录



- 10.1 决策树原理
- 10.2 迭代二叉树三代ID3算法
- 10.3 决策树C4.5算法
- 10.4 CART算法



10.1 决策树原理

10.2 迭代二叉树三代ID3算法

10.3 决策树C4.5算法

10.4 CART算法

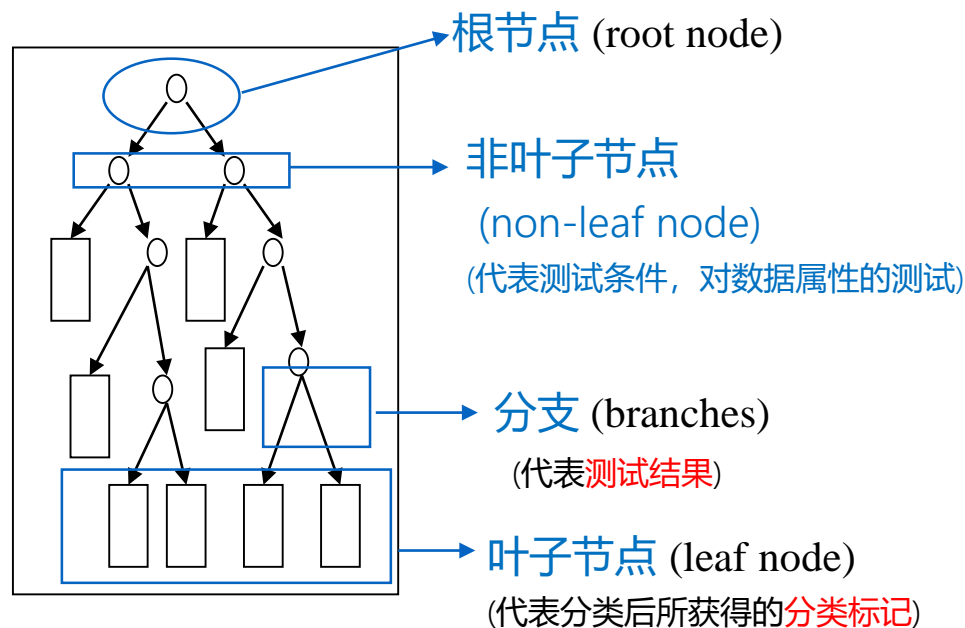
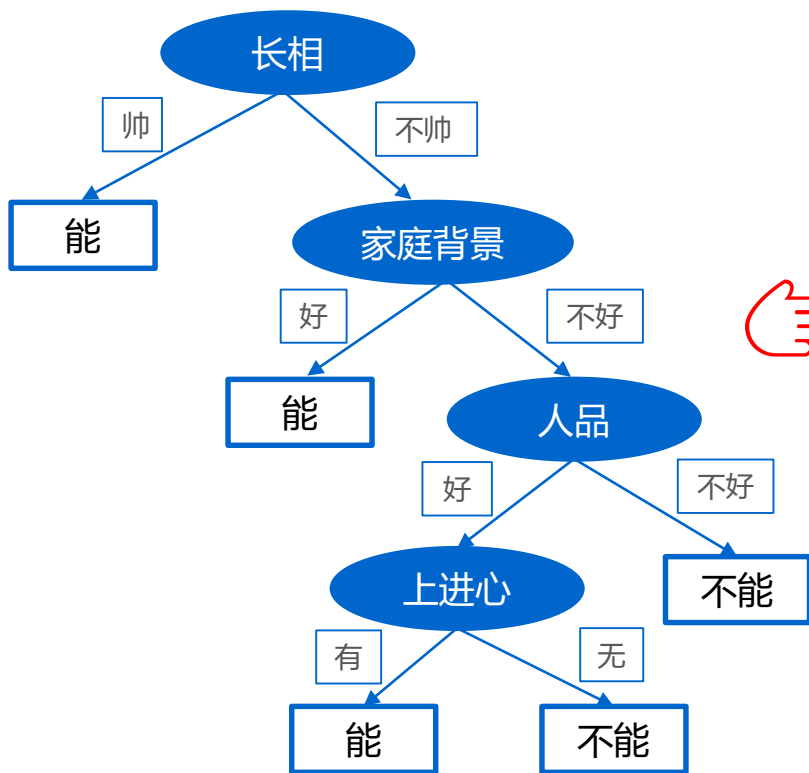
分类算法是利用训练样本集获得分类函数即分类模型(分类器)，从而实现将数据集中的样本划分到各个类中。分类模型通过学习训练样本中属性集与类别之间的潜在关系，并以此为依据对新样本属于哪一类进行预测。

- 假定公司收集了右表数据，那么对于任意给定的客户（测试样例），你能预测这位客户是属于“买”车的那一类，还是属于“不买”车的那一类？
- 另外，你需要多少有关这位客人的信息才能回答这个问题？

	年龄	有工作	有房子	信誉	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

决策树通过把数据样本分配到某个叶子结点来确定数据集中样本所属的分类

1. 决策树的基本原理



- 决策树由决策结点、分支和叶子组成。
 - 决策树中最上面的结点为根结点，是整个决策树的开始。每个分支是一个新的决策结点或者是树的叶子。
- 决策树：从训练数据中学习得出一个树状结构的模型。

- 决策树通过做出一系列决策（选择）来对数据进行划分，类似于针对一系列问题进行选择。
- 决策树的决策过程就是从根节点开始，遍历每个节点，测试待分类项中对应的特征属性，并按照其取值选择输出分支，直到叶子节点，将叶子节点的存放的类别作为决策结果。
- 每个决策结点代表一个问题或者决策，通常对应待分类对象的特征属性。
- 每个叶子结点代表一种可能的分类结果。
- 利用决策树进行分类，利用若干个变量来判断属性的类别，所以决策树属于判别模型。

🤖 决策树算法是一种归纳分类算法，它通过对训练集的学习，**挖掘（归纳）出有用的规则**，用于对新数据进行预测。

🤖 决策树属于**监督学习**方法。

🤖 决策树归纳的基本算法是**贪心算法**，自顶向下来构建决策树。

🤖 贪心算法：在每一步选择中都采取在当前状态下最好/优的选择。

🤖 在决策树的生成过程中，分割方法，也就是**属性选择的度量**，是关键。

决策树必须解决的问题

- ❖ 每一个分支节点的分叉条件是什么
- ❖ 节点处的分支数应该是几个？
- ❖ 如何确定某节点处应该测试哪个属性？
- ❖ 何时可以令某节点成为叶子节点？
- ❖ 如何使一个过大的树变小，如何“剪枝”？
- ❖ 如果叶子节点仍不“纯”，如何给它赋类别标记？
- ❖ 缺损的数据如何处理？
- ❖ 什么时候停止分叉？

决策树的数据准备与整理

❖ Data cleaning

删除/减少noise，补填missing values

❖ Data transformation

数据标准化（data normalization）

数据归纳（generalize data to higher-level）

例如：年龄归纳为老、中、青三类

控制每个属性的可能值不超过七种（最好不超过五种）

❖ Relevance analysis

对于与问题无关的属性：删

对于属性的可能值大于七种又不能归纳的属性：删

2. 决策树的特点

优点:

- 🧐 推理过程容易理解，计算简单，可解释性强。
- 🧐 比较适合处理有缺失属性的样本。
- 🧐 可自动忽略目标变量没有贡献的属性变量，也为判断属性变量的重要性，减少变量的数目提供参考。

缺点:

- 🧐 容易造成过拟合，需要采用剪枝操作。
- 🧐 忽略了数据之间的相关性。
- 🧐 对于各类别样本数量不一致的数据，信息增益会偏向于那些更多数值的特征。

3. 决策树的三种算法

- 🤖 1966年, Hunt, Marin和Stone 研制了CLS学习系统, 用于学习单个概念。
- 🤖 1979年, J.R. Quinlan 给出ID3算法, 并在1983年和1986年对ID3 进行了总结和简化, 使其成为决策树学习算法的典型。
- 🤖 1986年, Schlimmer 和Fisher 对ID3进行改进, 在每个可能的决策树节点创建缓冲区, 使决策树可以递增式生成, 得到ID4算法。
- 🤖 1988年, Utgoff 在ID4基础上提出了ID5学习算法, 进一步提高了效率。
- 🤖 1993年, Quinlan 进一步发展了ID3算法, 改进成C4.5算法。
- 🤖 另一类决策树算法为CART, 与C4.5不同的是, CART的决策树由二元逻辑问题生成, 每个树节点只有两个分枝, 分别包括学习实例的正例与反例。

🧐 建立决策树的关键，在当前状态下**选择哪个属性**作为分类依据。

🧐 根据不同的目标函数，建立决策树主要有一下三种算法：

🧐 ID3(Iterative Dichotomiser)、

🧐 C4.5、

🧐 CART(Classification And Regression Tree)。

算法	支持模型	树结构	特征选择	连续值处理	缺失值处理	剪枝	特征属性多次使用
ID3	分类	多叉树	信息增益	不支持	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益率	支持	支持	支持	不支持
CART	分类 回归	二叉树	基尼指数 均方差	支持	支持	支持	支持



10.1 决策树原理

10.2 迭代二叉树三代ID3算法

10.3 决策树C4.5算法

10.4 CART算法

迭代二叉树3代算法 (ID3: Iterative Dichotomiser 3)

🧐 ID3 算法最早是由罗斯昆 (J. Ross Quinlan) 于1975年提出的一种决策树构建算法，算法的核心是“信息熵”，期望信息越小，信息熵越大，样本纯度越低。

🧐 ID3 算法是以信息论为基础，以信息增益为衡量标准，从而实现对数据的归纳分类。

🧐 ID3 算法计算每个属性的信息增益，并选取具有最高增益的属性作为给定的测试属性。

1. ID3 算法的步骤

1. 初始化特征集合和数据集合；
2. 计算数据集合信息熵；
3. 对所有特征（属性）：1) 计算所有分类值的熵；2) 取当前特征的平均信息熵；3) 计算当前特征的增益；
4. 选择信息增益最大的特征作为当前决策节点；
5. 更新数据集合和特征集合（删除上一步使用的特征，并按照特征值来划分不同分支的数据集合）；
6. 重复 3，4 两步，若子集值包含单一特征，则为分支叶子节点。

什么是最好的特征？带来最大信息增益的特征。

一个分叉导致的信息增益越大, 则代表这次分叉提升的纯度越高。

当分到不能再分的时候, 此时的节点就是我们要的叶子节点。但是每个叶子节点并不都是熵为0, 叶子节点的表达形式就是以该节点占比较大的数据的类（标签）为结果的。

2. 信息增益的算法

- 🧐 设训练数据集为 D ， $N=|D|$ 表示其样本容量，即样本个数；
- 🧐 设有 K 个类 C_k , $k = 1, 2, \dots, K$ ， $|C_k|$ 为属于类 C_k 的样本个数；
- 🧐 特征 A 有 n 个不同的取值 $\{a_1, a_2, \dots, a_n\}$ 根据特征 A 的取值将 D 划分为 n 个子集 D_1, D_2, \dots, D_n ， $|D_i|$ 为 D_i 的样本个数；
- 🧐 记子集 D_i 中属于类 C_k 的样本集合为 D_{ik} ， $|D_{ik}|$ 为 D_{ik} 的样本个数。

输入：训练数据集 D 和特征 A

输出：特征 A 对训练数据集 D 的信息增益 $g(D, A)$

1. 计算数据集 D 的经验条件熵 $H(D)$

$$H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

$$H(D_i) = - \sum_{i=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

2. 计算特征 A 对数据集 D 的经验条件熵 $H(D/A)$

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

3. 计算信息增益（等价于训练数据集中类与特征的互信息）

$$g(D, A) = H(D) - H(D|A)$$

- 公式中用的输入样本数据在所有样本中出现的概率，如果数据集中重复的样本越多，则该样本的信息越小。
- 数据越纯，则数据的信息熵越小。
- 信息增益的定义是数据分叉前的信息熵减去分叉后的信息熵。
- 一个分叉导致的信息增益越大，则代表这次分叉提升的纯度越高。

平均信息熵计算

🤖 我们以某银行的个人信贷信息为例。
希望通过所给的训练数据学习一个贷款审核的决策树，用于对未来的贷款申请者进行分类，即当新的客户提出贷款申请时，根据申请人的特征利用决策树决定是否批准贷款申请。

数据如右表

数量	是	否	信息熵
15	9	6	0.971

$$\begin{aligned}
 H(D) &= - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|} \\
 &= - \frac{9}{15} \log_2 \frac{9}{15} - \frac{6}{15} \log_2 \frac{6}{15} = 0.971
 \end{aligned}$$

	年龄 A1	有工作 A2	有房子 A3	信誉 A4	类别 D
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

按年龄特征计算

年龄	数量	是	否	信息熵
青年	5	2	3	0.971
中年	5	3	2	0.971
老年	5	4	1	0.722

A_1	n=3
A_2	n=2
A_3	n=2
A_4	n=3

$$H(D|A_1 = \text{青年}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

$$H(D|A_1 = \text{中年}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$

$$H(D|A_1 = \text{老年}) = -\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0.722$$

按其他特征计算

$$H(D|A_2 = \text{是}) = -\frac{5}{5}\log_2\frac{5}{5} - 0 = 0$$

$$H(D|A_3 = \text{否}) = -\frac{3}{9}\log_2\frac{3}{9} - \frac{6}{9}\log_2\frac{6}{9} = 0.918$$

$$H(D|A_4 = \text{好}) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.918$$

	年龄	有工作	有房子	信誉	类别
	A1	A2	A3	A4	D
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

条件熵的计算:

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) \quad n \text{ 是特征 } A \text{ 的取值个数}$$

年龄 A_1 特征取值为青年、中年和老年三个类别，各有5个样本，所以计算 A_1 的条件熵:

$$\begin{aligned} H(D|A_1) &= \sum_{i=1}^3 \frac{|D_i|}{|D|} H(D_i) \\ &= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722 = 0.888 \end{aligned}$$

信息增益的计算:

$$\begin{aligned} g(D, A_1) &= H(D) - H(D|A_1) \\ &= 0.971 - 0.888 \\ &= 0.083 \end{aligned}$$

$$g(D, A_1 = \text{老年}) = H(D) - H(D|A_1 = \text{老年}) = 0.971 - 0.722 = 0.249$$

	年龄 A1	有工作 A2	有房子 A3	信誉 A4	类别 D
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

同理，计算其余特征的信息增益 $g(D, A_2)$ 、 $g(D, A_3)$ 和 $g(D, A_4)$ 。分别为：

$$\begin{aligned} g(D, A_2) &= H(D) - \left[\frac{5}{15} H(D_1) + \frac{10}{15} H(D_2) \right] \\ &= 0.971 - \left[\frac{5}{15} \times 0 + \frac{10}{15} \left(-\frac{4}{10} \log_2 \frac{4}{10} - \frac{6}{10} \log_2 \frac{6}{10} \right) \right] \\ &= 0.971 - 0.647 = 0.324 \end{aligned}$$

$$\begin{aligned} g(D, A_3) &= H(D) - \left[\frac{6}{15} H(D_1) + \frac{9}{15} H(D_2) \right] \\ &= 0.971 - \left[\frac{6}{15} \times 0 + \frac{9}{15} \left(-\frac{3}{9} \log_2 \frac{3}{9} - \frac{6}{9} \log_2 \frac{6}{9} \right) \right] \\ &= 0.971 - 0.551 = 0.420 \end{aligned}$$

$$g(D, A_4) = 0.971 - 0.608 = 0.363$$

最后，比较特征的信息增益，由于特征 A_3 (有房子)的信息增益值最大，所以选择 A_3 作为最优特征。

	年龄 A1	有工作 A2	有房子 A3	信誉 A4	类别 D
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

3. ID3算法实现流程

D : 样本集合, A : 样本属性集合

1. 创建根节点Root;
2. 如果 D 中的元素属于同一类别, 则为单节点树, 标记为该类别标号, 返回Root;
3. 如果 A 为空, 则为单节点树, 标记为 D 中元素数最大的类别标号, 返回Root;
4. 否则, 计算 A 中各特征对 D 的信息增益, 选择信息增益最大的特征: $A_i \leftarrow A$ 中分类能力最强的特征;
5. 如果 A_i 的信息增益小于给定阈值, 为单节点树, 标记为 D 中元素数最大的类别标号, 返回Root;
6. 否则, 将 D 中的元素根据 A_i 的特征 (可能取值 a_j) 分成若干子集, 将 D_i 中元素数最大的类别标号作为标记, 构建子节点, 由节点和其子节点构成子树, 返回Root;
7. 若 D_i 为空, 则在新分支下加入一个叶子节点, 属性标记为 D 中最普遍的类别;
8. 否则在这个分支下加入一个子节点ID3($D_i, A-\{A_i\}$)。

- ❑ 迭代构建决策树：得到原始数据集，然后基于最好的属性值划分数据集，由于特征值可能多于两个，因此可能存在大于两个分支的数据集划分，第一次划分之后，数据将被向下传递到树分支的下一个节点，在此节点再次划分数据，因此可以使用递归的原则处理数据集。
- ❑ 递归结束的条件是：程序完全遍历所有划分数据集的属性，或者每个分支下的所有实例都具有相同的分类，如果所有实例具有相同的分类，则得到一个叶子节点或者终止块，任何到达叶子节点的数据必然属于叶子节点的分类。

4. ID3算法小结

🤖 ID3算法的基本思想：是以信息熵为度量，用于决策树节点的属性选择，每次优先选取信息量最多的属性，亦即使熵值变为最小的属性，以构造一颗熵值下降最快的决策树，到叶子节点处的熵值为0。此时，每个叶子节点对应的实例集中的实例属于同一类。

🤖 理想的决策树有三种：

- (1) 叶子结点数最少；
- (2) 叶子结点深度最小；
- (3) 叶子结点数最少且叶子结点深度最小。

🤖 然而，有人已经证明了要找到这种最优的决策树是NP难题。

🤖 因此，决策树优化的目的就是要找到尽可能趋向于最优的决策树。

ID3算法的缺点

- 🙄 ID3 没有剪枝策略，容易过拟合；
- 🙄 信息增益准则对可取值数目较多的特征有所偏好，类似“编号”的特征其信息增益接近于 1；
- 🙄 只能用于处理离散分布的特征；
- 🙄 没有考虑缺失值。



10.1 决策树原理

10.2 迭代二叉树三代ID3算法

10.3 决策树C4.5算法

10.4 CART算法

决策树C4.5算法

C4.5 算法是 Ross 对 ID3 算法的改进。

🧐 用**信息增益率**来选择属性。ID3选择属性用的是子树的信息增益，而C4.5用的是**信息增益率**。

🧐 在决策树构造过程中进行**剪枝**。

🧐 对**非离散数据**也能处理。

🧐 能够对**不完整数据**进行处理。

信息增益率

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}$$

其中, $H_A(D) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$, n 是特征 A 的取值个数

$$\begin{aligned} g(D, A_1 = \text{老年}) &= H(D) - H(D|A_1 = \text{老年}) \\ &= 0.971 - 0.7219 = 0.2491 \end{aligned}$$

$$\begin{aligned} g_R(D, A_1 = \text{老年}) &= \frac{g(D, A_1 = \text{老年})}{H_A(D)} \\ &= \frac{0.2491}{- \sum_{i=1}^n \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}} \\ &= \frac{0.2491}{-\frac{9}{15} \log_2 \frac{9}{15} - \frac{6}{15} \log_2 \frac{6}{15}} = 0.2565 \end{aligned}$$

	年龄 A1	有工作 A2	有房子 A3	信誉 A4	类别 D
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

备注: 信息增益 $g(D, A) = H(D) - H(D|A)$

决策树C4.5算法流程

D: 样本集合，**A**: 样本属性集合

1. 创建根节点Root;
2. 如果**D**中的元素属于同一类别，则为单节点树，标记为该类别标号，返回Root;
3. 如果**A**为空，则为单节点树，标记为**D**中元素数最大的类别标号，返回Root;
4. 否则，计算**A**中各特征对**D**的信息增益率，选择信息增益率最大的特征 **A_i** ，即**A**中**分类能力最强的特征**;
5. 如果 **A_i** 的信息增益小于给定阈值，为单节点树，该节点标记为**D**中元素数最大的类别标号，返回Root;
6. 否则，对 **A_i** 的每个可能特征取值 a_j ，以 **$A_i=a_j$** 将**D**分成若干子集 **D_i** ，将 **D_i** 中元素数最大的类别标号作为标记，构建子节点，由节点和其子节点构成树，返回Root;
7. 对节点 **i** ，以 **D_i** 为训练集，以 **$A-\{A_i\}$** 为特征集，逐步递归1~5，得到子树 **T_i** ，返回Root。

1. C4.5的剪枝

过拟合的原因：

决策树算法增长树的每一个分支的深度，直到恰好能对训练样例比较完美地分类。为了尽可能正确分类训练样本，节点的划分过程会不断重复直到不能再分，这样就可能对训练样本学习的“太好”了，把训练样本的一些特点当做所有数据都具有的一般性质。实际应用中，当数据中有噪声或训练样例的数量太少以至于不能产生目标函数的有代表性的采样时，该策略可能会遇到困难，从而导致过拟合。

通过剪枝处理去掉一些分支来降低过拟合的风险。

剪枝的基本策略有“**预剪枝**”（prepruning）和“**后剪枝**”（post-pruning）

2. 预剪枝 (prepruning)

预剪枝不仅可以降低过拟合的风险，而且还可以减少训练时间，但是另一方面它是基于“贪心”策略，会带来欠拟合风险。

剪枝策略

在节点划分前来确定是否继续增长，及早停止增长。

主要方法有：

在算法中提前设置**超参数**，当决策树分叉出的子节点满足条件时就停止分叉，最终的子节点作为叶子节点。

- 可设置的超参数有：叶子节点中最少数据，少于这个数据就不分了；所有节点特征都已分裂；节点划分前准确率比划分后准确率高；决策树一共可以分多少层，达到层数停止分裂；决策树子节点的数量等等。

3. 后剪枝

- 🧐 在已经生成的决策树上进行剪枝，从而得到简化版的剪枝决策树。
- 🧐 后剪枝决策树通常比预剪枝决策树保留了更多的分支。一般情况下，后剪枝的欠拟合风险更小，泛化性能往往优于预剪枝决策树。
- 🧐 后剪枝是对决策树不做限制，生成一个完全生长的过拟合的树，使用现有的测试集一点一点的减掉叶子节点暴露新的叶子节点，然后迭代测试。

- 🤖 后剪枝的算法包括Reduced-Error Pruning (REP, 错误率降低剪枝)和Pessimistic Error Pruning (PEP, 悲观剪枝).
- 🤖 采用的悲观剪枝方法, 用递归的方式从低往上针对每一个非叶子节点, 评估用一个最佳叶子节点去代替这棵子树是否有益。如果剪枝后与剪枝前相比其错误率是保持或者下降, 则这棵子树就可以被替换掉。
- 🤖 后剪枝需要用训练集不断的实验如何保留分支, 并且要自底向上的对树种的所有非叶子节点进行逐一考察, 因此训练时间开销比未剪枝决策树和预剪枝决策树都要大很多。

4. C4.5的缺点



剪枝策略可以再优化；



C4.5 用的是多叉树，用二叉树效率更高；



C4.5 只能用于分类；



C4.5 使用的熵模型拥有大量耗时的对数运算，连续值还有排序运算；



C4.5 在构造树的过程中，对数值属性值需要按照其大小进行排序，从中选择一个分割点，所以只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时，程序无法运行。




10.1 决策树原理

10.2 迭代二叉树三代ID3算法


10.3 决策树C4.5算法

10.4 CART算法

CART算法


 Classification and Regression Tree(CART)算法采用的是一种二分循环分割的方法，每次都把当前样本集划分为两个子样本集，使生成的决策树的结点均有两个分支，构造一个二叉树。

 CART算法既可以用于创建分类树，也可以用于创建回归树，两者在构建的过程中稍有差异。

 如果目标变量是离散的，称为分类树，常用基尼指数来选择属性。

 如果目标变量是连续的，称为回归树，常用均方差来选择属性。

CART与ID3的区别



二元划分：


二叉树不易产生数据碎片，精确度往往也会高于多叉树。



CART由两部分组成：

决策树生成

决策树剪枝



剪枝：

用预剪枝或后剪枝对训练集生长的树进行剪枝



树的建立：

- 如果目标变量是标称的，并且是具有两个以上的类别，则CART可能考虑将目标类别合并成两个超类别（双化）；
- 如果目标变量是连续的，则CART算法找出一组基于树的回归方程来预测目标变量。

1. 基尼指数

- 对于给定样本集合 D ， C_k 是 D 中属于第 k 类的样本子集， K 为 D 中类别的个数，其基尼指数为：

$$Gini(D) = \sum_{k=1}^K \frac{|C_k|}{|D|} \left(1 - \frac{|C_k|}{|D|}\right) = 1 - \sum_{k=1}^K \left(\frac{|C_k|}{|D|}\right)^2$$

- 如果样本集合 D 根据特征 A 是否取某个可能值 a 被分割成 D_1 和 D_2 两部分，即

$$D_1 = \{x \in D | A(x) = a\}, \quad D_2 = D - D_1$$

- 那么在特征 A 条件下， D 的基尼指数为

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

$$\text{➤ } K = 2, Gini(D_i) = 2 \frac{|D_{ik}|}{|D_i|} \left(1 - \frac{|D_{ik}|}{|D_i|}\right)$$

- $Gini(D)$ 表示集合 D 的不确定性。 $Gini(D, A)$ 表示经过 $A=a$ 分割后集合 D 的不确定性。基尼指数越大，样本集合的不确定性就越大。

$$Gini(D, A_1 = \text{青年}) = \frac{5}{15} \times \left(2 \times \frac{2}{5} \times \left(1 - \frac{2}{5} \right) \right) + \frac{10}{15} \times \left(2 \times \frac{7}{10} \times \left(1 - \frac{7}{10} \right) \right) = 0.44$$

$$Gini(D, A_1 = \text{中年}) = 0.48$$

$$Gini(D, A_1 = \text{老年}) = 0.44$$

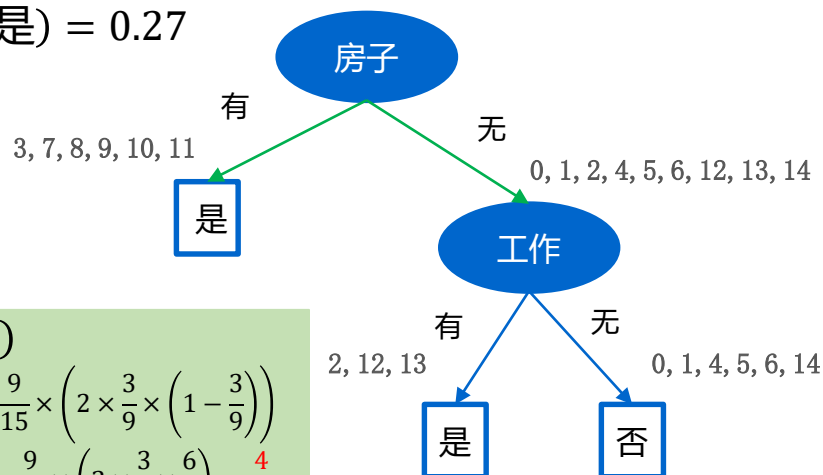
$$Gini(D, A_2 = \text{是}) = 0.32$$

$$Gini(D, A_4 = \text{好}) = 0.47$$

$$Gini(D, A_4 = \text{非常好}) = 0.36$$

$$Gini(D, A_4 = \text{一般}) = 0.32$$

$$Gini(D, A_3 = \text{是}) = 0.27$$



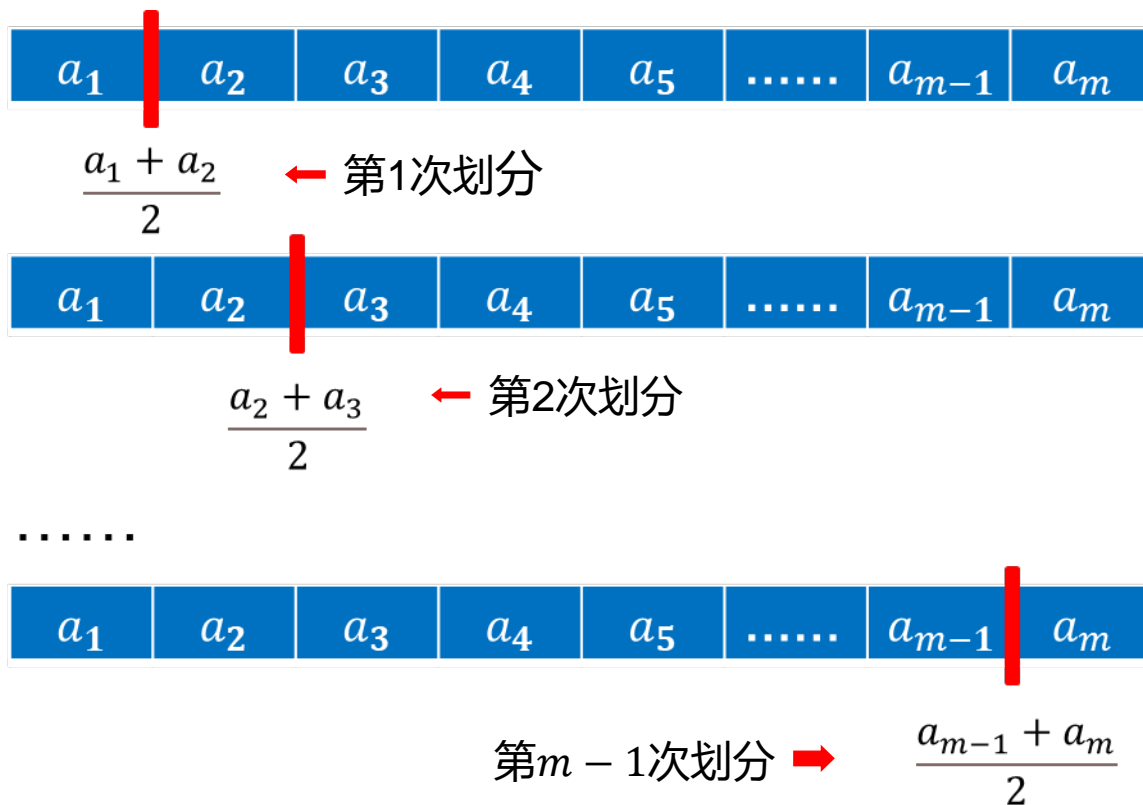
$$Gini(D, A_3 = \text{是}) = \frac{6}{15} \times \left(\frac{6}{6} \times \left(1 - \frac{6}{6} \right) \right) + \frac{9}{15} \times \left(2 \times \frac{3}{9} \times \left(1 - \frac{3}{9} \right) \right) = 0 + \frac{9}{15} \times \left(2 \times \frac{3}{9} \times \frac{3}{9} \right) = \frac{9}{15} \times \left(2 \times \frac{3}{9} \times \frac{6}{9} \right) = \frac{4}{15}$$

	年龄 A1	有工作 A2	有房子 A3	信誉 A4	类别 D
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

连续特征处理

具体思路： m 个样本的连续特征 A 有 m 个， $a_1, a_2, a_3, \dots, a_{m-1}, a_m$ 从小到大排列，取相邻两样本值的平均数做划分点，其中第 $m-1$ 个划分点 T_m 表示为：

$T_m = \frac{a_{m-1} + a_m}{2}$ 。分别计算以这 $m-1$ 个点作为二元分类点时的基尼系数。选择基尼指数最小的点为该连续特征的二元离散分类点。



比如取到的基尼指数最小的点为 a_t ，则小于 a_t 的值为类别1，大于 a_t 的值为类别2，这样就做到了连续特征的离散化，接着采用基尼指数的大小来度量特征的各个划分点。

离散特征处理

具体思路：

假设特征 A 有 m 个离散值。分类标准是：每一次将其中一个特征分为一类，其他非该特征分为另一类。依照这个标准遍历所有分类情况，计算每个分类下的基尼指数，最后选择最小的作为最终的特征划分。



← 第1次划分



← 第2次划分

.....



第 m 次划分 →

比如第1次取 $\{a_1\}$ 为类别1，那么剩下的特征 $\{a_2, a_3, \dots, a_{m-1}, a_m\}$ 为类别2，由此遍历，第 m 次取 $\{a_m\}$ 为类别1，那么剩下的特征 $\{a_1, a_2, a_3, \dots, a_{m-1}\}$ 为类别2。

2. CART算法（分类树）

输入：训练数据集 D ，停止计算条件

输出：CART决策树

从根节点开始，递归对每个结点操作：

1. 设结点数据集为 D ，对每个特征 A ，对其每个值 a ，根据样本点对 $A=a$ 的测试为是或否，将 D 分为 D_1 ， D_2 ，计算 $A=a$ 的基尼指数；
2. 在所有的特征 A 以及所有可能的切分点 a 中，选择基尼指数最小的特征和切分点，将数据集分配到两个子结点中；
3. 对两个子结点递归调用1，2步骤；
4. 生成CART树。

3. CART算法-回归树

用均方差来选择属性


- 对于连续值的处理，CART 分类树采用基尼系数的大小来度量特征的各个划分点。
- 对于任意划分特征 A ，对应的任意划分点 s 两边划分成的数据集 D_1 和 D_2 ，求出使 D_1 和 D_2 各自集合的均方差最小，同时 D_1 和 D_2 的均方差之和最小所对应的特征和特征值划分点。


表达式为：

$$\min_{a,s} [\min_{c_1} \sum_{x_i \in D_1} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in D_2} (y_i - c_2)^2]$$

其中， c_1 为 D_1 数据集的样本输出均值， c_2 为 D_2 数据集样本输出均值。

预测方式

 对于决策树建立后做预测的方式，上面讲到了 CART 分类树采用叶子节点里概率最大的类别作为当前节点的预测类别。

 而回归树输出不是类别，它采用的是用最终叶子的均值或者中位数来预测输出结果。

4. CART剪枝

- CART算法采用一种“基于代价复杂度的剪枝”方法进行后剪枝。这种方法会生成一系列树，每个树都是通过将前面的树的某个或某些子树替换成一个叶子节点而得到的。这一系列树中的最后一棵树仅含一个用来预测类别的叶子节点。
- 然后用一种成本复杂度的度量准则来判断哪棵子树应该被一个预测类别值的叶子节点所代替。
- 这种剪枝方法需要使用一个单独的测试数据集来评估所有的树，根据它们在测试数据集上的分类性能，选出最佳的树。

CART剪枝具体流程:

- (1) 计算每一个结点的条件熵
- (2) 递归的从叶子节点开始往上遍历, 减掉叶子节点, 然后判断损失函数的值是否减少; 如果减少, 则将父节点作为新的叶子节点
- (3) 重复(2), 直到完全不能剪枝。

决策树差异总结

🤖 **划分标准的差异**: ID3 使用信息增益偏向特征值多的特征, C4.5 使用信息增益率克服信息增益的缺点, 偏向于特征值小的特征, CART 使用基尼指数克服 C4.5 需要求 \log 的巨大计算量, 偏向于特征值较多的特征;

🤖 **使用场景的差异**: ID3 和 C4.5 都只能用于分类问题, CART 可以用于分类和回归问题; ID3 和 C4.5 是多叉树, 速度较慢, CART 是二叉树, 计算速度很快;



样本数据的差异： ID3 只能处理离散数据且缺失值敏感，C4.5 和 CART 可以处理连续性数据且有多种方式处理缺失值；从样本量考虑的话，小样本建议用 C4.5，大样本建议用 CART。C4.5 处理过程中需对数据集进行多次遍历排序，处理成本耗时较高，而 CART 本身是一种大样本的统计方法，小样本处理下泛化误差较大；



样本特征的差异： ID3 和 C4.5 层级之间只使用一次特征，CART 可多次重复使用特征；



剪枝策略的差异： ID3 没有剪枝策略，C4.5 是通过悲观剪枝策略来修正树的准确性，而 CART 是通过代价复杂度剪枝。

实际使用决策树做预测需要以下过程：

收集数据：可以使用任何方法。比如想构建一个相亲系统，我们可以从介绍人那里，或者通过采访相亲对象获取数据。根据他们考虑的因素和最终的选择结果，就可以得到一些供我们利用的数据了。

准备数据：收集完的数据，我们要进行整理，将这些所有收集的信息按照一定规则整理出来，并排版，方便我们进行后续处理。

分析数据：可以使用任何方法，决策树构造完成之后，我们可以检查决策树图形是否符合预期。

训练算法：这个过程也就是构造决策树，同样也可以说是决策树学习，就是构造一个决策树的数据结构。

测试算法：使用经验树计算错误率。当错误率达到了可接收范围，这个决策树就可以投放使用了。

使用算法：此步骤可以使用适用于任何监督学习算法，而使用决策树可以更好地理解数据的内在含义。

类别：既能分类 又能回归

定义：可以用分类问题的训练数据（特征，二义树）

定义和组成

- 根节点：（含根节点）：决策树根——判断条件（特征）
- 叶子节点：预测结果（类）

决策过程

根据某一维分类问题的特征，按照其值划分分支，依次向下，到达叶子节点

- 1 在输入的训练数据（假设m个样本）
- 2 利用每个样本属性的类别
- 3 人为选取一些特征（即决策条件）
- 4 为每个训练样本对应所需要的特征位置赋值——数据特征

将通过上面的1-4步得到的训练数据输入公式计算，训练算法通过一定的原理，决定每个节点的属性值，然后按照决策属性值从根节点开始，依次向下遍历

决策树的构造过程是一个迭代过程

特点

- 每次迭代中，采用不同特征作为分裂点（分类特征），将样本数据划分成不同的类别
- 迭代分裂特征的目标：在保证一个分裂子集中的样本数尽可能小，类别

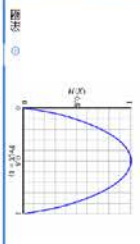
核心：以信息增益为度量，选择分裂后信息增益最大的特征进行分裂

提示：随着迭代不断进行，数据中信息量不断减少，信息熵降低，信息增益大

一个随机变量x的信息熵 $Entropy(x) = -\sum_{i=1}^n p_i \log_2(p_i)$

全部样本集合S的信息熵 $Entropy(S) = -\sum_{i=1}^n p_i \log_2(p_i)$

$$-\log_2(1-x) - (1-x) \log_2(1-x)$$



熵：给定一个事件以后不确定性或混乱的程度

信息增益

定义：将样本集合S基于特征T进行分裂后获得的信息增益

InformationGain

$$Gain(T) = Entropy(S) - \sum_{i=1}^n p_i Entropy(S_i)$$

算法特点

- 1 对数据多的特征分裂，分裂后的结果信息增益高；分裂结果相同，则信息增益越大
- 2 不能处理连续特征或区间的特征

出现原因

为了得到D算法的不足（1），采用信息增益——用信息熵来衡量不确定性

分裂信息

作用：使用信息熵来衡量不确定性

SplitInformation

$$SplitInformation(T) = -\sum_{i=1}^n p_i \log_2 \frac{p_i}{n}$$

信息增益

作用：防止信息增益过大，信息增益过大，信息增益小，信息增益小的问题

Gain Ratio

$$GainRatio(T) = \frac{Gain(T)}{SplitInformation(T)}$$

其他优点

对属性值和对数据量和数据分布要求敏感的问题

算法特点

SplitInformation(T) → 0, GainRatio(T) → ∞

解决方法

采用交叉验证原理，对每个特征先计算信息增益，在信息增益最大的情况下，才采用信息增益作为分裂标准

特点

每次选择Gain系数最小的特征作为最优特征

CART算法

CART的构造过程就是求Gain系数（Gain Coefficient）

区别

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵

用于计算一个数据集的信息熵，对每个特征先计算信息增益，在信息增益最大的情况下，才采用信息增益作为分裂标准

熵增益

每次选择Gain系数最小的特征作为最优特征

Gini系数

Gini Coefficient

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

熵增益

如果特征信息增益，而特征信息增益与k个特征，则选择一个特征转化为k个特征，对每一个特征信息增益不是取这个值分 Yes 和 No

参考文献:

1) 【白话机器学习】算法理论+实战之决策树

https://mp.weixin.qq.com/s/4_nuHfspT7Iry_PTfRWTWw

2) 机器学习实战（三）——决策树

https://blog.csdn.net/jiaoyangwm/article/details/79525237?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc_relevant_default&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc_relevant_default&utm_relevant_index=1

谢谢!



上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

上海交通大学

本课件制作过程中，多处引用了国内外同行的网页、教材、以及课件PPT的内容或图片，没有随处标注，特此说明，并在此向各位作者表示感谢！