

—武大本科生课程



第8讲 决策树

(Lecture 8 Decision Tree)

武汉大学计算机学院机器学习课程组

2023.03

第6章 决策树

内容目录

6.1 决策树的概念

6.2 决策树学习

6.3 结点最佳划分特征的选择

6.4 决策树的生成

6.4.1 决策树的生成：ID3算法

6.4.2 决策树的生成：CART算法

6.5 决策树的剪枝(*：了解)

6.6 决策树Python程序示例

小结

6.1 决策树的概念

- 例子1 套用俗语，决策树分类的思想类似于找对象。现想象一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的母女对话：

女儿：那人多大年纪了？

母亲：26。

女儿：长的帅不帅？

母亲：挺帅的。

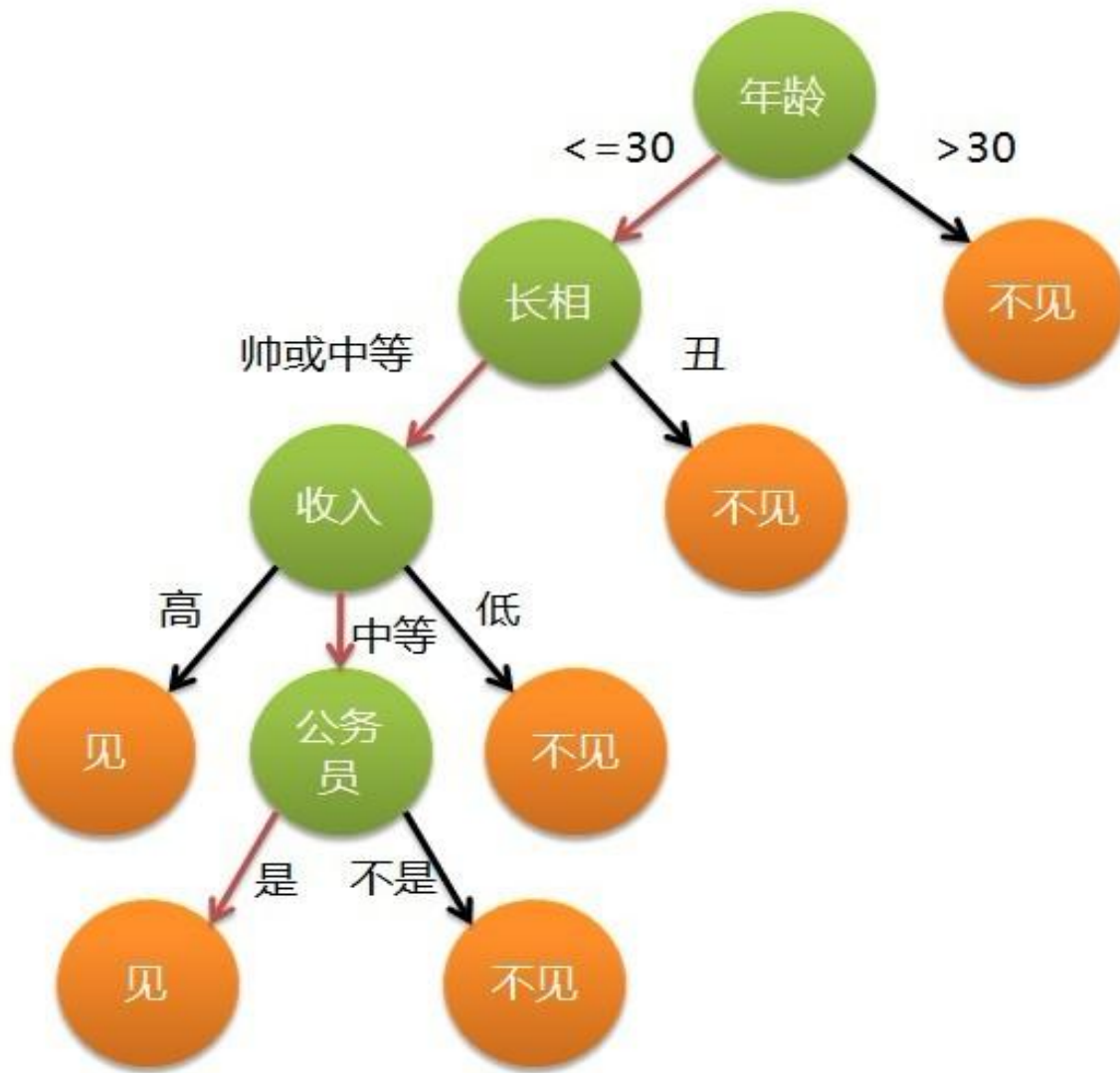
女儿：收入高不？

母亲：不算很高，中等情况。

女儿：是公务员不？

母亲：是，在税务局上班呢。

女儿：那好，我去见见。



“是否与候选男朋友见面”的决策示意图

决策树(Decision tree)的原理是通过对一系列问题进行if-then的推导，最终实现决策。

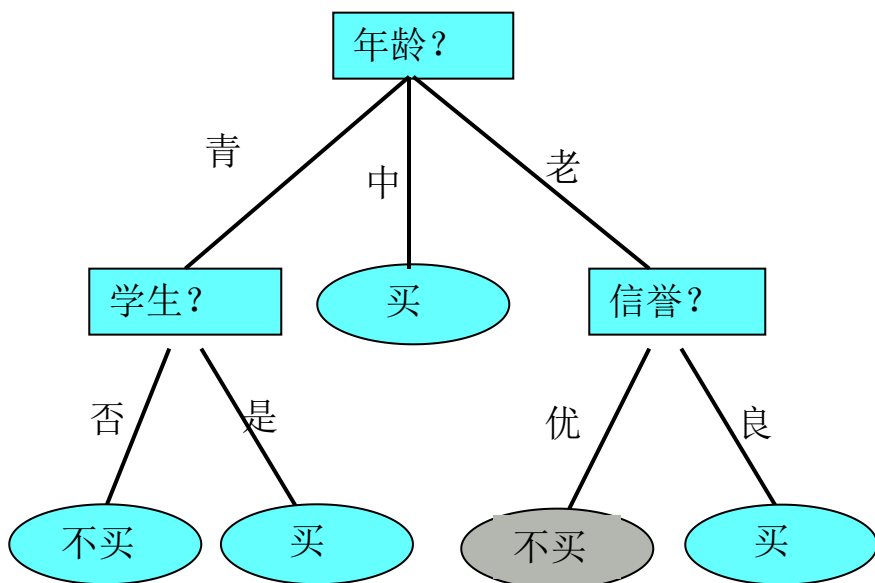
■ **例子2** 假定某公司收集了下表数据，那么对于任意给定的客人(测试样例)，你能帮助公司将这位客人归类吗？

即：你能预测这位客人是属于“买”计算机的那一类，还是属于“不买”计算机的那一类？

又：你需要多少有关这位客人的信息才能回答这个问题？

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

谁在买计算机？



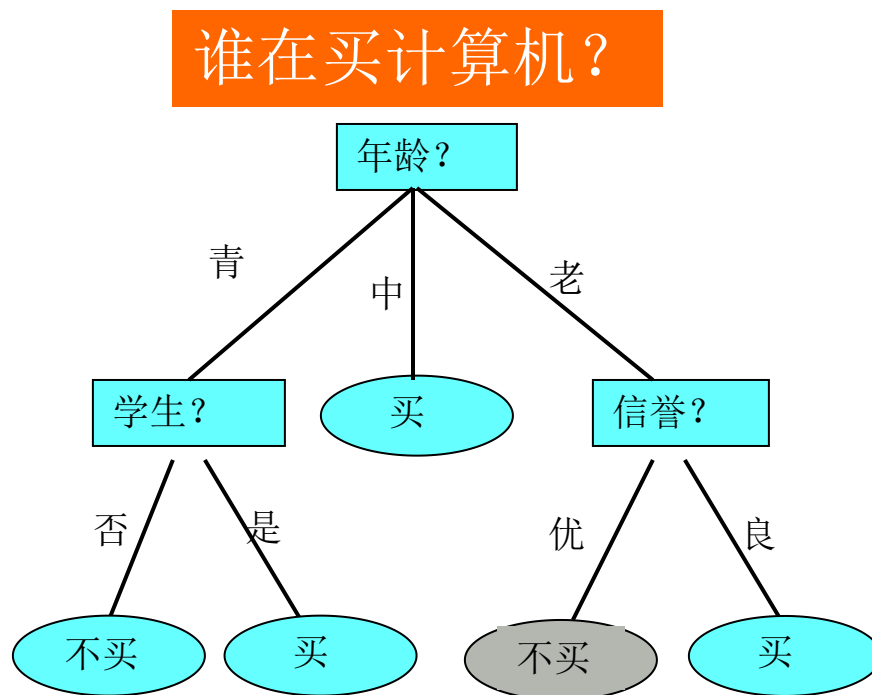
计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

决策树分类思想

- **基于规则的思维模式：**人类在进行某些取舍决策或事物判断时，一般会基于一些规则，而且这些规则往往是潜在的，只在遇到具体的判断条件时，才会显式地、具体地表现出来。
- 前例“是否与候选男朋友见面”的判断过程，以及医生根据生理指标与临床诊断，判断一个人是否患有某种疾病的过程等，都是基于规则的思维模式的例子，本质上是通过询问一系列不同方面的问题，根据这些问题的回答最终达成序列式组合判断。
- **决策树模型：**一种与上述基于问答式规则进行事物性质或类别判断过程与思维逻辑相同的计算模型。由于很多判断规则多数时候是潜在的，我们能了解的往往是据以判断的属性取值和最终的判断结果，因此，**使用计算机建立决策树模型**，与建立其他分类器模型一样，**需要研究如何从这些训练数据中，提取潜在判断规则的方法。**

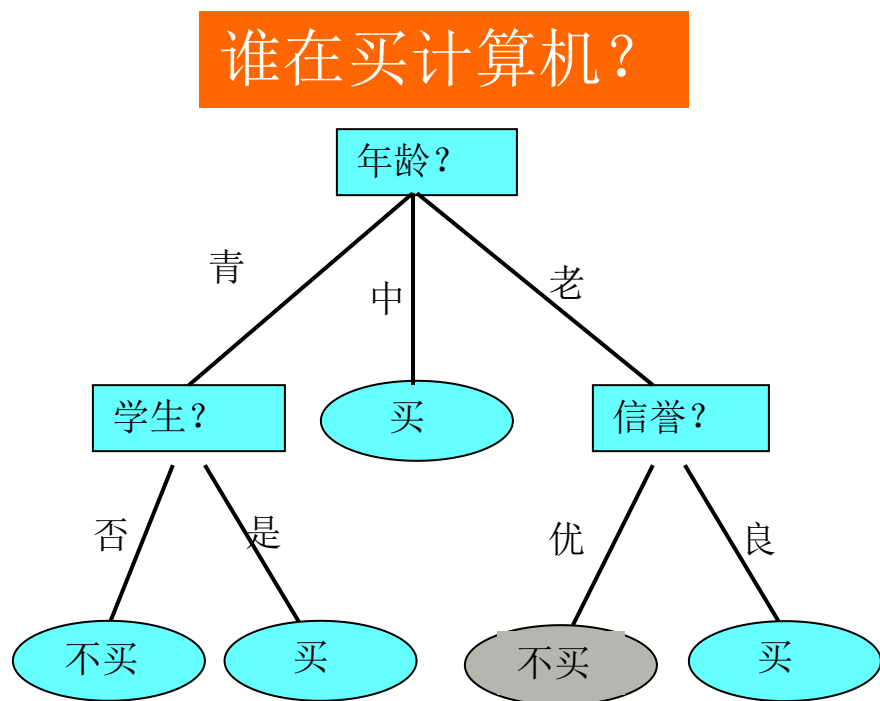
- **分类决策树的建立：**把样本的特征空间（输入空间）划分为这样一组互斥的单元，其中每个单元中样本的类别要尽可能地“单纯、一致”，其中优势样本类别就设定为“单元的代表类”。

决策树定义 分类决策树模型是一种描述对实例进行分类的树形结构。决策树由结点 (node) 和有向边 (directed edge) 组成。结点有两种类型：内部结点 (internal node) 和叶结点 (leaf node)。内部结点表示一个特征或属性，叶结点表示一个类。



决策树的表示

决策树的基本组成部分：决策结点、分支和叶子。



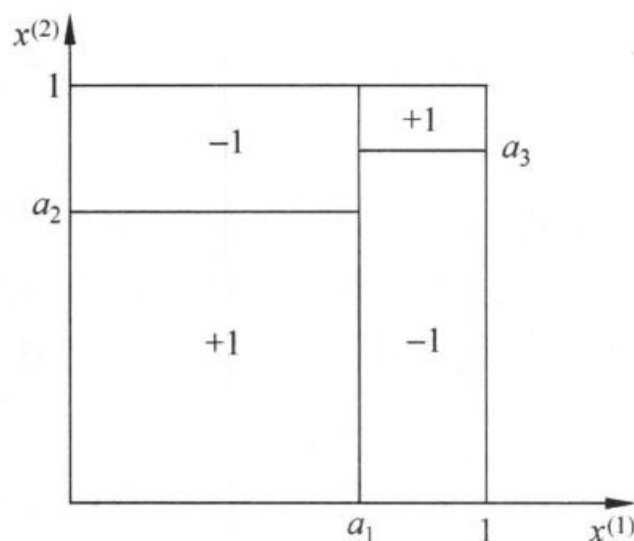
- 决策树中最上面的结点称为根结点。是整个决策树的开始。每个分支是一个新的决策结点，或者是树的叶子。
- 每个决策结点代表一个问题或者决策。通常对应待分类对象的属性。
- 每个叶结点代表一种可能的分类结果。

在沿着决策树从上到下的遍历过程中，在每个结点都有一个测试。对每个结点上问题的不同测试输出导致不同的分支，最后会达到一个叶子结点。这一过程就是利用决策树进行分类的过程，利用若干个变量来判断属性的类别。

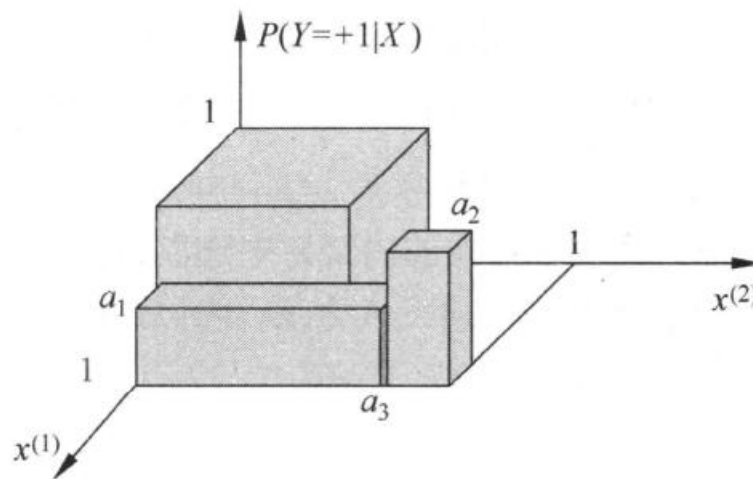
- **分类决策树的判决：**用训练好的决策树对新实例进行分类，是从**根结点**开始的。按决策时确定的规则，对新实例的**某一特征进行测试**，根据测试结果，将实例分配到合适的**子结点**。在子结点上，递归进行同样的测试与子结点分配，直至达到叶结点。实例的类别被“预测”为叶结点的“**单元代表类**”。
- 从结点的定义和形成过程看，这些最后用于分类的单元 (即决策树的叶结点) 代表的其实就是前面的“问答式进入规则”的分类结果，落入叶结点的训练样本就是满足这些进入规则、有确定分类结果的样本。

可以将决策树看成一个**if-then规则的集合**。将决策树转换成if-then规则的过程是这样的：由决策树的根结点到叶结点的每一条路径构建一条规则；路径上内部结点的特征对应着规则的条件，而叶结点的类对应着规则的结论。**决策树的路径或其对应的if-then规则集合具有一个重要的性质：互斥并且完备**。这就是说，每一个实例都被一条路径或一条规则所覆盖，而且只被一条路径或一条规则所覆盖。这里所谓覆盖是指实例的特征与路径上的特征一致或实例满足规则的条件。

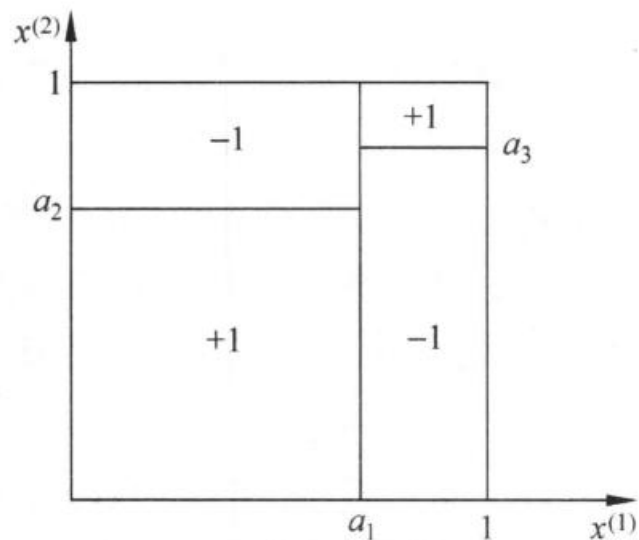
决策树还表示给定特征条件下类的**条件概率分布**。这一条件概率分布定义在特征空间的一个**划分(partition)**上。将特征空间划分为互不相交的**单元(cell)**或**区域(region)**，并在每个单元定义一个类的概率分布就构成了一个**条件概率分布**。决策树的一条路径对应于划分中的一个单元。决策树所表示的条件概率分布由各个单元给定条件下类的条件概率分布组成。假设 **X** 为表示特征的随机变量， **Y** 为表示类的随机变量，那么这个条件概率分布可表示为 **$P(Y|X)$** 。 **X** 取值于给定划分下单元的集合， **Y** 取值于类的集合。各叶结点(单元)上的条件概率往往偏向某一个类，即属于某一类的概率较大。决策分类时将该结点的实例强行分到条件概率大的那一类中。



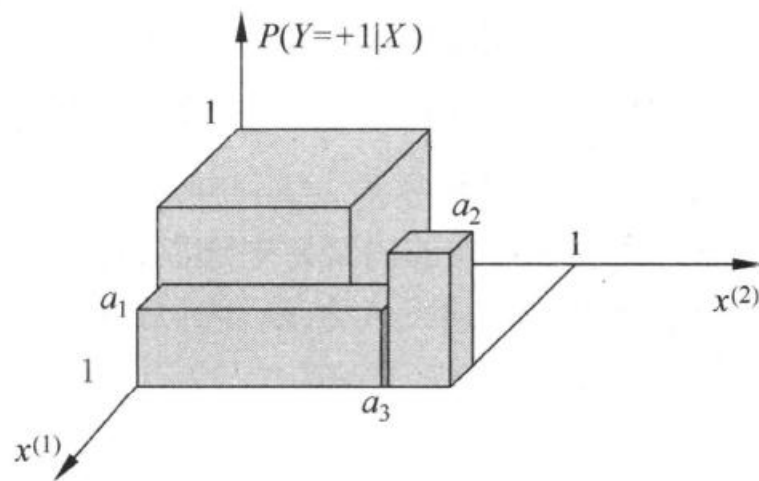
(a) 特征空间划分



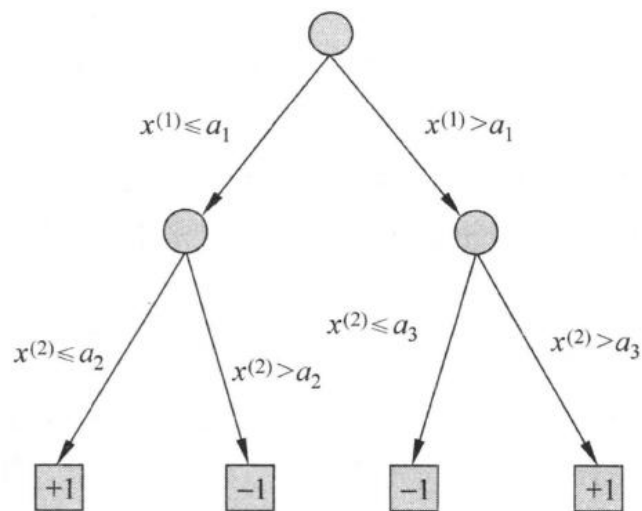
(b) 条件概率分布



(a) 特征空间划分



(b) 条件概率分布



(c) 连续取值特征的决策树构建

图 决策树对应于条件概率分布

6.2 决策树学习

假设给定训练数据集

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

其中, $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$ 为输入实例 (特征向量), n 为特征个数, $y_i \in \{1, 2, \dots, K\}$ 为类标记, $i = 1, 2, \dots, N$, N 为样本容量。决策树学习的目标是根据给定的训练数据集构建一个决策树模型, 使它能够对实例进行正确的分类。

决策树学习本质上是从训练数据集中归纳出一组分类规则。与训练数据集不相矛盾的决策树 (即能对训练数据进行正确分类的决策树) 可能有多, 也可能一个也没有。我们需要的是一个与训练数据矛盾较小的决策树, 同时具有很好的泛化能力。从另一个角度看, 决策树学习是由训练数据集估计条件概率模型。基于特征空间划分的类的条件概率模型有无穷多个。我们选择的条件概率模型应该不仅对训练数据有很好的拟合, 而且对未知数据有很好的预测。

决策树学习用损失函数表示这一目标。如下所述, 决策树学习的损失函数通常是正则化的极大似然函数。决策树学习的策略是以损失函数为目标函数的最小化。

决策树学习的算法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得对各个子数据集有一个最好的分类的过程。这一过程对应着对特征空间的划分，也对应着决策树的构建。开始，构建根结点，将所有训练数据都放在根结点。选择一个最优特征，按照这一特征将训练数据集分割成子集，使得各个子集有一个在当前条件下最好的分类。如果这些子集已经能够被基本正确分类，那么构建叶结点，并将这些子集分到所对应的叶结点中去；如果还有子集不能被基本正确分类，那么就对这些子集选择新的最优特征，继续对其进行分割，构建相应的结点。如此递归地进行下去，直至所有训练数据子集被基本正确分类，或者没有合适的特征为止。最后每个子集都被分到叶结点上，即都有了明确的类。这就生成了一棵决策树。

可以看出，**决策树学习算法包含特征选择、决策树的生成与决策树的剪枝过程。**由于决策树表示一个条件概率分布，所以**深浅不同的决策树对应着不同复杂度的概率模型。****决策树的生成对应于模型的局部选择，决策树的剪枝对应于模型的全局选择。**决策树的生成只考虑局部最优，相对地，决策树的剪枝则考虑全局最优。

决策树学习常用的算法有ID3、C4.5与CART。下面**结合ID3、CART算法**分别叙述决策树学习的特征选择、决策树的生成和剪枝过程。

6.3 结点最佳划分特征的选择

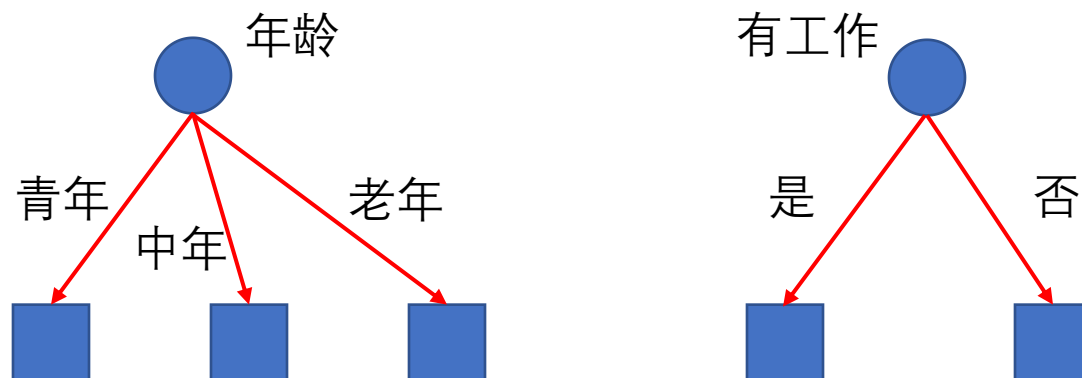
- 结点**最佳划分特征的选择原则**：相对于选取其他特征进行划分，根据这个最佳特征对当前结点的样本数据进行划分、形成子结点后，各个子结点的“类别分布更加集中、更加单纯”。这样的特征是对当前结点最具分类能力的特征。注意：不同结点的最佳划分特征是不同的。
- 使用最佳结点划分特征可以提高决策树的学习效率。如果利用一个特征进行分类的结果与随机分类的结果没有很大差别，则称这个特征是没有分类能力的，扔掉这样的特征对决策树的学习精度影响不大。通常特征选择的准则是**信息增益**。

例6.1 表6.1 是一个由 15 个样本组成的贷款申请训练数据。数据包括贷款申请人的 4 个特征（属性）：第 1 个特征是年龄，有 3 个可能值：青年，中年，老年；第 2 个特征是有工作，有 2 个可能值：是，否；第 3 个特征是有自己的房子，有 2 个可能值：是，否；第 4 个特征是信贷情况，有 3 个可能值：非常好，好，一般。表的最后一列是类别，是否同意贷款，取 2 个值：是，否。

表6.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

- 希望通过所给的训练数据学习一个贷款申请的决策树，用以对未来的贷款申请进行分类， 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。
- **特征选择**是决定用哪个特征来划分特征空间。



不同的特征选择顺序，形成不同的决策树

结点划分特征的选择依据：信息增益

在信息论与概率统计中，熵 (Entropy) 是表示随机变量不确定性的度量。

设 X 是一个取有限个值的离散随机变量，其概率分布为

$$P(X = x_i) = p_i, \quad i = 1, 2, \dots, n$$

则随机变量 X 的熵定义为

$$H(X) = - \sum_{i=1}^n p_i \log p_i \quad (6.1)$$

在式 (6.1) 中，若 $p_i = 0$ ，则定义 $0 \log 0 = 0$ 。通常，式 (6.1) 中的对数以 2 底或以 e 为底（自然对数），这时熵的单位分别称作比特 (bit) 或纳特 (nat)。由定义可知，熵只依赖于 X 的分布，而与 X 的取值无关，所以也可将 X 的熵记作 $H(p)$ ，

即

$$H(p) = - \sum_{i=1}^n p_i \log p_i \quad (6.2)$$

结点划分特征的选择依据：信息增益

补充：熵与信息量的概念

熵的概念最早源自物理学中的热力学，热力学第二定律和第三定律都是与熵有关的定律。在热力学中，**熵的物理意义是一个物理体系混乱程度的度量**，是表征物质状态的参量之一。

熵是一个宏观物理量，是构成物理体系的大量微观粒子集体表现出来的一种物理性质。在统计热力学中，**设物理体系的微观状态为 Ω** ，**则物理体系的熵可表示为 $S=k\log(\Omega)$**

其中， **$k=1.3807 \times 10^{-23} \text{J} \cdot \text{K}^{-1}$** ，**为玻尔兹曼常量**，体系微观状态 Ω 是大量质点的体系经统计规律而得到的热力学概率。

物理体系的熵，可以通过宏观系统的热交换进行计算。根据克劳修斯的定义，如果物理体系加热过程引起的变化是可逆的，则系统的熵增为

$$DS = \left(\frac{dQ}{T} \right)_r$$

其中， T 表示体系的热力学温度， Q 表示熵增过程中加入体系的热量， r 表示物理过程可逆。

热力学第二定律一般又称为熵增定律，表明在自然过程中，一个孤立系统的熵或混乱程度不会减少。

结点划分特征的选择依据：信息增益

补充：熵与信息量的概念

信息论中的熵也称为香农熵(Shannon Entropy)或信息熵(Information Entropy),前者以信息论奠基人香农的名字命名。它反映了一个概率分布的随机性程度,或者说它包含的信息量的大小。

首先考虑随机变量取某一特定值时所包含的信息量大小。假设随机变量 X 取某个值 x 的概率为 $p(x)$,取这个值的概率很小但它却发生了,则包含的信息量大。考虑两个随机事件:

(1)一年之内载人火箭登陆火星;

(2)武汉明天要下雨。

显然前者包含的信息量要大于后者,因为前者的概率远小于后者但却发生了。

如果定义一个函数 $h(x)$ 来描述随机变量取值 x 时信息量的大小,则 $h(x)$ 应为 $p(x)$ 的单调减函数。

符合单调减要求的函数并不唯一,需要进一步缩小范围。假设有两个相互独立的随机变量 X 和 Y ,它们分别取值为 x 和 y 的概率为 $p(x)$ 和 $p(y)$ 。由于相互独立,因此它们的联合概率为:

$$p(x,y) = p(x)p(y)$$

由于随机变量 X 和 Y 相互独立,因此它们取值为 (x,y) 的信息量应该是 X 取值为 x 且 Y 取值为 y 的信息量之和,即正式成立:

$$h(x,y) = h(x) + h(y)$$

因此,要求 $h(x)$ 能把 $p(x)$ 的乘法运算转化为加法运算,满足此要求的基本函数是对数函数。

结点划分特征的选择依据：信息增益

补充：熵与信息量的概念

可以把**信息量**定义为：

$$h(x) = -\log p(x)$$

对数函数前面加上负号是因为对数函数是增函数，而我们要求 $h(x)$ 是 $p(x)$ 的减函数。另外。由于 $0 \leq p(x) \leq 1$ ，因此 $\log p(x) \leq 0$ ，加上负号之后恰好可以保证信息量非负。信息量与随机变量所取的值 x 本身无关，只与它取值的概率有关。对数底数取值并不重要，在通信领域中通常以2为底(以**比特bit**为单位)，在机器学习常以 e (自然对数)为底(以**纳特nat**为单位)。

以上只考虑了随机变量取某个值时所包含的信息量，随机变量可以取多个值，因此需要计算它取所有各种值所包含的信息量。随机变量取每个值有一个概率，因此可以计算它取各个值时信息量的数学期望，这个均值就是熵。

对于离散型随机变量 X ，假设该变量取值有 n 种情况 $x_i (i=1, 2, \dots, n)$ ，则**熵**定义为

$$H(p) = E_p[-\log p(x)] = -\sum_{i=1}^n p_i \log p_i$$

这里约定 $p_i = p(x_i)$

可以证明，随机变量越接近均匀分布(随机性越强)，熵越大；反之则熵越小。

设有随机变量 (X, Y) ，其联合概率分布为

$$P(X = x_i, Y = y_j) = p_{ij}, \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$$

条件熵 $H(Y|X)$ 表示在已知随机变量 X 的条件下随机变量 Y 的不确定性。随机变量 X 给定的条件下随机变量 Y 的条件熵 (conditional entropy) $H(Y|X)$ ，定义为 X 给定条件下 Y 的条件概率分布的熵对 X 的数学期望

$$H(Y|X) = \sum_{i=1}^n p_i H(Y|X = x_i) \quad (6.3)$$

这里， $p_i = P(X = x_i)$ ， $i = 1, 2, \dots, n$ 。

信息增益(gain)定义

特征 A 对训练数据集 D 的信息增益 $g(D, A)$ ，定义为集合 D 的经验熵 $H(D)$ 与特征 A 给定条件下 D 的经验条件熵 $H(D|A)$ 之差，即

$$g(D, A) = H(D) - H(D|A) \quad (6.4)$$

一般地，熵 $H(Y)$ 与条件熵 $H(Y|X)$ 之差称为互信息 (mutual information)。决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

补充：互信息的概念

互信息定义了两个随机变量的依赖程度。对于两个离散型随机变量 X 和 Y ，它们之间的互信息定义为：

$$I(X, Y) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

其中 $p(x, y)$ 为 X 和 Y 的联合概率， $p(x)$ 和 $p(y)$ 分别为 X 和 Y 的边缘概率。互信息反映了联合概率 $p(x, y)$ 与边缘概率之积 $p(x)p(y)$ 的差异程度。如果两个随机变量相互独立，则 $p(x, y) = p(x)p(y)$ ，因此它们越接近于相互独立，则 $p(x, y)$ 和 $p(x)p(y)$ 的值越接近。

根据定义，互信息具有对称性：

$$I(X, Y) = I(Y, X)$$

互信息性质：

$$I(Y, X) = H(Y) - H(Y | X)$$

决策树学习应用信息增益准则选择特征。给定训练数据集 D 和特征 A , 经验熵 $H(D)$ 表示对数据集 D 进行分类的不确定性。而经验条件熵 $H(D|A)$ 表示在特征 A 给定的条件下对数据集 D 进行分类的不确定性。那么它们的差, 即信息增益, 就表示由于特征 A 而使得对数据集 D 的分类的不确定性减少的程度。显然, 对于数据集 D 而言, 信息增益依赖于特征, 不同的特征往往具有不同的信息增益。信息增益大的特征具有更强的分类能力。

根据信息增益准则的特征选择方法是: 对训练数据集 (或子集) D , 计算其每个特征的信息增益, 并比较它们的大小, 选择信息增益最大的特征。

设训练数据集为 D , $|D|$ 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, $\sum_{k=1}^K |C_k| = |D|$ 。设特征 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, $\sum_{i=1}^n |D_i| = |D|$ 。记子集 D_i 中属于类 C_k 的样本的集合为 D_{ik} , 即 $D_{ik} = D_i \cap C_k$, $|D_{ik}|$ 为 D_{ik} 的样本个数。于是信息增益的算法如下。

算法6.1 （信息增益的算法）

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 的经验熵 $H(D)$

$$H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|} \quad (6.5)$$

(2) 计算特征 A 对数据集 D 的经验条件熵 $H(D|A)$

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \quad (6.6)$$

(3) 计算信息增益

$$g(D, A) = H(D) - H(D|A) \quad (6.7)$$

例6.2 对表6.1 所给的训练数据集 D ，根据信息增益准则选择最优特征。

解 首先计算经验熵 $H(D)$ 。

$$H(D) = -\frac{9}{15} \log_2 \frac{9}{15} - \frac{6}{15} \log_2 \frac{6}{15} = 0.971$$

然后计算各特征对数据集 D 的信息增益。分别以 A_1, A_2, A_3, A_4 表示年龄、有工作、有自己的房子和信贷情况 4 个特征，则

(1)

$$\begin{aligned} g(D, A_1) &= H(D) - \left[\frac{5}{15} H(D_1) + \frac{5}{15} H(D_2) + \frac{5}{15} H(D_3) \right] \\ &= 0.971 - \left[\frac{5}{15} \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) + \right. \\ &\quad \left. \frac{5}{15} \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) + \frac{5}{15} \left(-\frac{4}{5} \log_2 \frac{4}{5} - \frac{1}{5} \log_2 \frac{1}{5} \right) \right] \\ &= 0.971 - 0.888 = 0.083 \end{aligned}$$

这里 D_1, D_2, D_3 分别是 D 中 A_1 (年龄) 取值为青年、中年和老年的样本子集。类似地，

(2)

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

(3)

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

(4)

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

最后，比较各特征的信息增益值。由于特征 A_3 （有自己的房子）的信息增益值最大，所以选择特征 A_3 作为最优特征。

6.4.1 决策树的生成：ID3算法

- ID3算法是一种经典的决策树学习算法，由Quinlan于1979年在悉尼大学提出。
- ID3算法主要针对特征选择问题。是决策树学习方法中最具影响和最为典型的算法。
- 当获取信息时，将不确定的内容转为确定的内容，因此信息伴着不确定性。
- ID3 算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地构建决策树。从根结点开始，对本结点计算所有可能的特征的信息增益，选择信息增益最大的特征作为本结点的划分特征，由该特征的不同取值建立子结点；再对子结点递归地调用以上方法，构建决策树；直到所有特征的信息增益均很小或没有特征可以选择为止。最后得到一棵决策树。ID3 相当于用极大似然法进行概率模型的选择。

算法6.2 (ID3 算法)

输入：训练数据集 D ，特征集 A 阈值 ε ；

输出：决策树 T 。

(1) 若 D 中所有实例属于同一类 C_k ，则 T 为单结点树，并将类 C_k 作为该结点的类标记，返回 T ；

(2) 若 $A = \emptyset$ ，则 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T ；

(3) 否则，按 算法6.2 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_g ；

(4) 如果 A_g 的信息增益小于阈值 ε ，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T ；

(5) 否则，对 A_g 的每一可能值 a_i ，依 $A_g = a_i$ 将 D 分割为若干非空子集 D_i ，将 D_i 中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树 T ，返回 T ；

(6) 对第 i 个子结点，以 D_i 为训练集，以 $A - \{A_g\}$ 为特征集，递归地调用步 (1)~步 (5)，得到子树 T_i ，返回 T_i 。

例6.3

对表6.1 的训练数据集，利用ID3 算法建立决策树。

解 利用 例6.2 的结果，由于特征 A_3 (有自己的房子) 的信息增益值最大，所以选择特征 A_3 作为根结点的特征。它将训练数据集 D 划分为两个子集 D_1 (A_3 取值为“是”) 和 D_2 (A_3 取值为“否”)。由于 D_1 只有同一类的样本点，所以它成为一个叶结点，结点的类标记为“是”。

对 D_2 则需从特征 A_1 (年龄)， A_2 (有工作) 和 A_4 (信贷情况) 中选择新的特征。计算各个特征的信息增益：

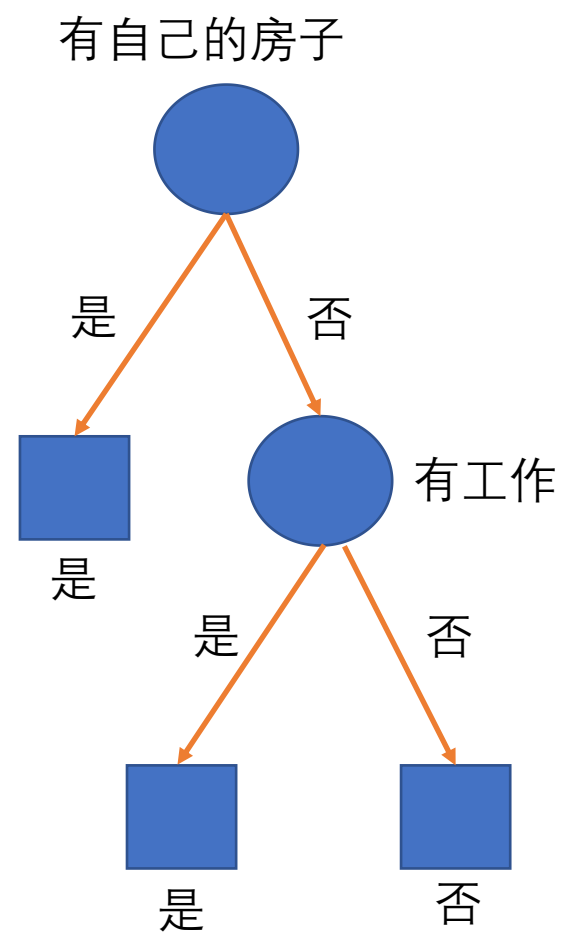
$$g(D_2, A_1) = H(D_2) - H(D_2|A_1) = 0.918 - 0.667 = 0.251$$

$$g(D_2, A_2) = H(D_2) - H(D_2|A_2) = 0.918$$

$$g(D_2, A_4) = H(D_2) - H(D_2|A_4) = 0.474$$

选择信息增益最大的特征 A_2 (有工作) 作为结点的特征。由于 A_2 有两个可能取值，从这一结点引出两个子结点：一个对应“是”(有工作)的子结点，包含 3 个样本，它们属于同一类，所以这是一个叶结点，类标记为“是”；另一个是对应“否”(无工作)的子结点，包含 6 个样本，它们也属于同一类，所以这这也是一个叶结点，类标记为“否”。

这样生成一棵如 图(a) 所示的决策树。该决策树只用了两个特征 (有两个内部结点)。



图(a) 例6.3中构建的决策树

6.4.2 决策树的生成：CART算法

- **分类与回归树** (classification and regression tree, CART) 模型由Breiman 等人于1984年提出，是一种应用广泛的决策树学习方法。CART由特征选择、树的生成及剪枝组成，既可以用于分类也可以用于回归。以下将用于分类与回归的树统称为决策树。
- CART 是在给定输入随机变量 X 条件下输出随机变量 Y 的条件概率分布的学习方法。**CART 决策树是二叉树**，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是“否”分支。这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在给定的输入特征条件下，输出的条件概率分布。
- **CART决策树的生成**就是递归地构建二叉决策树的过程。结点最优划分特征的选择使用基尼指数(Gini index)。

基尼指数(基尼不纯度)定义 分类问题中, 假设有 K 个类, 样本点属于第 k 类的概率为 p_k , 则**概率分布的基尼指数定义为**

$$\text{Gini}(p) = \sum_{k=1}^K p_k(1 - p_k) = 1 - \sum_{k=1}^K p_k^2 \quad (6.8)$$

对于二类分类问题, 若样本点属于第 1 个类的概率是 p , 则概率分布的基尼指数为

$$\text{Gini}(p) = 2p(1-p) \quad (6.9)$$

对于给定的样本集合 D , 其基尼指数为:

$$\text{Gini}(D) = 1 - \sum_{k=1}^K \left(\frac{|C_k|}{|D|} \right)^2 \quad (6.10)$$

这里, C_k 是 D 中属于第 k 类的样本子集, K 是类的个数。

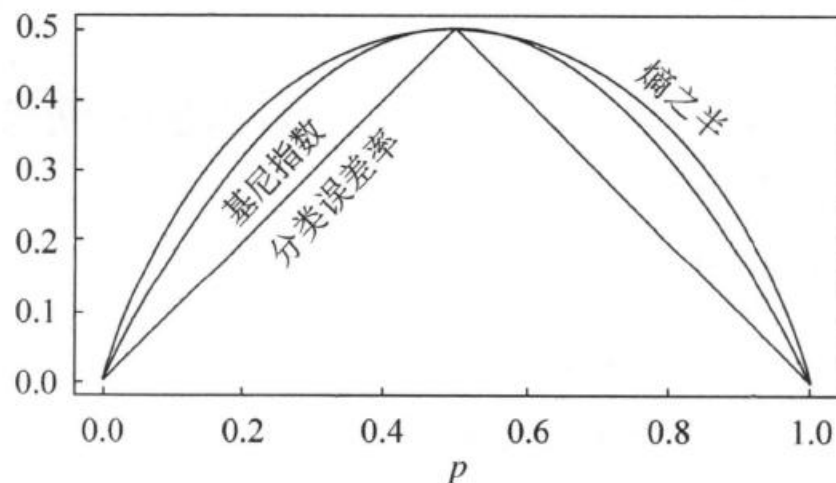
如果样本集合 D 根据特征 A 是否取某一可能值 a 被分割成 D_1 和 D_2 两部分, 即

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

则在特征 A 条件下, 集合 D 的基尼指数定义:

$$\text{Gini}(D, A) = \frac{|D_1|}{|D|} \text{Gini}(D_1) + \frac{|D_2|}{|D|} \text{Gini}(D_2) \quad (6.11)$$

基尼指数 $\text{Gini}(D)$ 表示集合 D 的不确定性, 基尼指数 $\text{Gini}(D, A)$ 表示经 $A = a$ 分割后集合 D 的不确定性。基尼指数值越大, 样本集合的不确定性也就越大, 这一点与熵相似。



二分类中 基尼指数、信息熵和分类误差率的关系

算法3 (CART 生成算法)

输入：训练数据集 D ，停止计算的条件；

输出：CART 决策树。

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

(1) 设结点的训练数据集为 D ，计算现有特征对该数据集的基尼指数。此时，对每一个特征 A ，对其可能取的每个值 a ，根据样本点对 $A = a$ 的测试为“是”或“否”将 D 分割成 D_1 和 D_2 两部分，利用式 (6.11) 计算 $A = a$ 时的基尼指数。

(2) 在所有的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点。依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去。

(3) 对两个子结点递归地调用 (1)，(2)，直至满足停止条件。

(4) 生成 CART 决策树。

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值（样本基本属于同一类），或者没有更多特征。

特征 A 条件集合 D 的基尼指数：

$$\text{Gini}(D, A) = \frac{|D_1|}{|D|} \text{Gini}(D_1) + \frac{|D_2|}{|D|} \text{Gini}(D_2) \quad (6.11)$$

例6.4 根据表6.1 所给训练数据集，应用CART 算法生成决策树。

解 首先计算各特征的基尼指数，选择最优特征以及其最优切分点。仍采用 例6.2 的记号，分别以 A_1, A_2, A_3, A_4 表示年龄、有工作、有自己的房子和信贷情况 4 个特征，并以 1, 2, 3 表示年龄的值为青年、中年和老年，以 1, 2 表示有工作和有自己的房子的值为是和否，以 1, 2, 3 表示信贷情况的值为非常好、好和一般。

求特征 A_1 的基尼指数：

$$\text{Gini}(p) = 2p(1-p)$$

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

$$\text{Gini}(D, A_1 = 2) = 0.48$$

$$\text{Gini}(D, A_1 = 3) = 0.44$$

特征 A 条件集合 D 的基尼指数：

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

由于 $\text{Gini}(D, A_1 = 1)$ 和 $\text{Gini}(D, A_1 = 3)$ 相等，且最小，所以 $A_1 = 1$ 和 $A_1 = 3$ 都可以选作 A_1 的最优切分点。

求特征 A_2 和 A_3 的基尼指数：

$$\text{Gini}(D, A_2 = 1) = 0.32$$

$$\text{Gini}(D, A_3 = 1) = 0.27$$

由于 A_2 和 A_3 只有一个切分点，所以它们就是最优切分点。

求特征 A_4 的基尼指数:

$$\text{Gini}(D, A_4 = 1) = 0.36$$

$$\text{Gini}(D, A_4 = 2) = 0.47$$

$$\text{Gini}(D, A_4 = 3) = 0.32$$

$\text{Gini}(D, A_4 = 3)$ 最小, 所以 $A_4 = 3$ 为 A_4 的最优切分点。

在 A_1, A_2, A_3, A_4 几个特征中, $\text{Gini}(D, A_3 = 1) = 0.27$ 最小, 所以选择特征 A_3 为最优特征, $A_3 = 1$ 为其最优切分点。于是根结点生成两个子结点, 一个是叶结点。对另一个结点继续使用以上方法在 A_1, A_2, A_4 中选择最优特征及其最优切分点, 结果是 $A_2 = 1$ 。依此计算得知, 所得结点都是叶结点。

对于本问题, 按照 CART 算法所生成的决策树与按照 ID3 算法所生成的决策树完全一致。

6.5 决策树的剪枝(*: 了解)

- 决策树生成算法递归地产生决策树，直到不能继续下去为止。这样产生的树往往对训练数据的分类很准确，但对未知的测试数据的分类却没有那么准确，即出现过拟合现象。过拟合的原因在于学习时过多地考虑如何提高对训练数据的正确分类，从而构建出过于复杂的决策树，解决该问题的办法考虑决策树的复杂度，对已生成的决策树进行简化。
- 在决策树学习中将已生成的树进行简化的过程称为**剪枝(pruning)**。具体地说，剪枝从已生成的树上裁掉一些子树或叶结点，并将其根结点或父结点作为新的叶结点，从而简化分类树模型。
- 决策树学习的剪枝算法(此略)。

6.6 决策树Python程序示例

- 示例：使用Sklearn库自带的红酒数据集生成一棵决策树。

决策树Python程序示例：使用红酒数据集展示决策树的构建

```
In [1]: #Filename: ch6_wine_dt.ipynb
import numpy as np
#导入画图工具
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#导入tree模型和数据集加载工具
from sklearn import tree, datasets
#导入数据集拆分工具
from sklearn.model_selection import train_test_split
```

为了方便数据可视化展示，我们仍然选取数据集中样本的前两个特征。

```
In [2]: #选取数据集的前两个特征
wine = datasets.load_wine()
X = wine.data[:, :2]
y = wine.target
#将数据集拆分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
In [3]: #设置决策树分类器最大深度为1
clf = tree.DecisionTreeClassifier(max_depth=1)
#拟合训练数据集
clf.fit(X_train, y_train)
```

```
Out[3]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=1, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')
```

决策树分类器返回max_depth参数是决策树的深度(所问的问题数量越多，决策树的深度越深)

```
In [4]: clf.score(X_test, y_test)
```

```
Out[4]: 0.5555555555555556
```


数据可视化，观察分类器表现

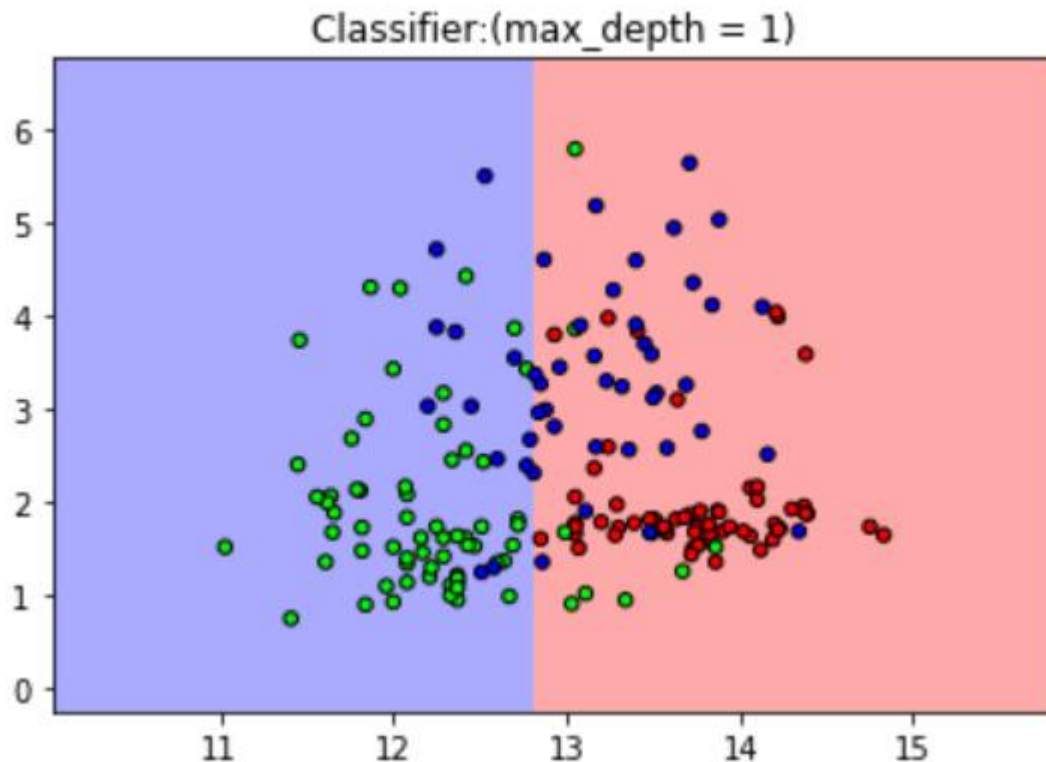
```
In [5]: #定义图像中分区的颜色和散点的颜色
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

#分别用样本的两个特征值创建图像和横轴和纵轴
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                     np.arange(y_min, y_max, .02))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

#给每个分类中的样本分配不同的颜色
Z = Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

#用散点把样本表示出来
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolor='k', s=20)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Classifier: (max_depth = 1)")

plt.show()
```



结果分析：显然。最大深度max_depth=1时分类器表现不是太好，分类器只分了两类。

下面加大深度，设置max_depth=3，看结果是否有变化。

```
In [6]: #设置决策树分类器最大深度为3
        clf2 = tree.DecisionTreeClassifier(max_depth=3)
        clf2.fit(X_train, y_train)
```

```
Out[6]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=3, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')
```

```
In [7]: clf2.score(X_test, y_test)
```

```
Out[7]: 0.7555555555555555
```

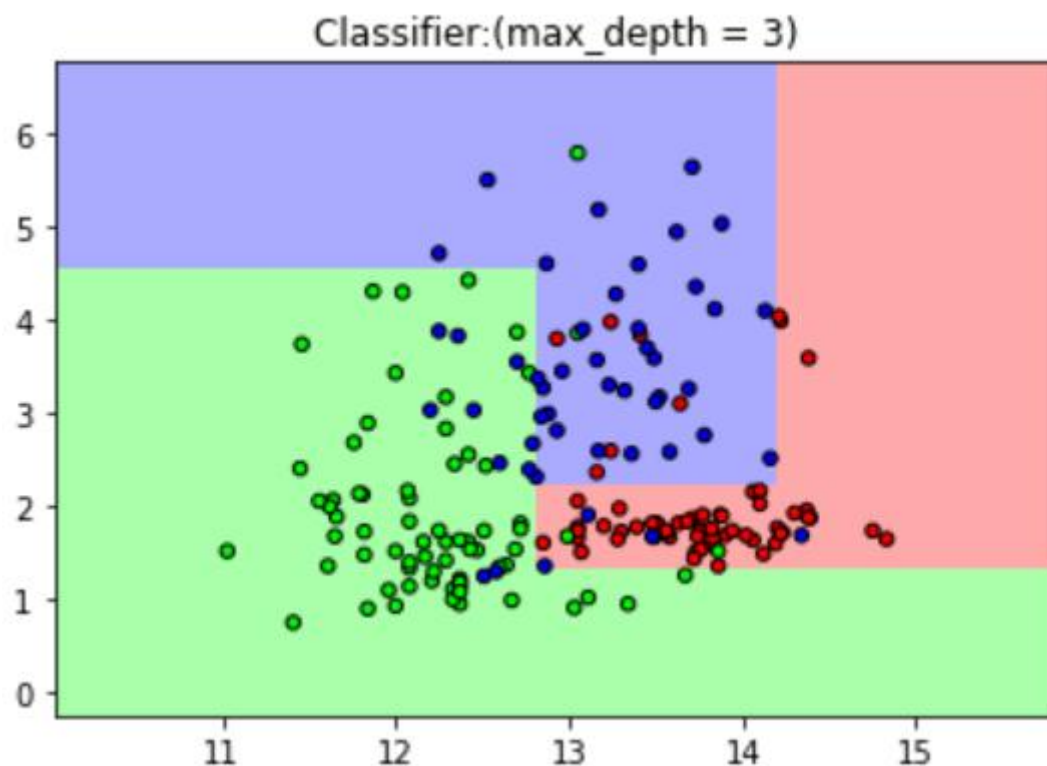
```
In [8]: #定义图像中分区的颜色和散点的颜色
        cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
        cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

        #分别用样本的两个特征值创建图像和横轴和纵轴
        x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
        y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
        xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                              np.arange(y_min, y_max, .02))
        Z = clf2.predict(np.c_[xx.ravel(), yy.ravel()])

        #给每个分类中的样本分配不同的颜色
        Z = Z.reshape(xx.shape)
        plt.figure()
        plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

        #用散点把样本表示出来
        plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolor='k', s=20)
        plt.xlim(xx.min(), xx.max())
        plt.ylim(yy.min(), yy.max())
        plt.title("Classifier:(max_depth = 3)")

        plt.show()
```



结果分析：可以看出，当最大深度max_depth=3时分类器能够进行3分类识别，而且大数数据点进了进入了正确的分类，当然还有一小部分数据点是错误的。接下来进一步调整max_depth=5，看结果有何变化。

```
In [9]: #设置决策树分类器最大深度为5
clf3 = tree.DecisionTreeClassifier(max_depth=5)
clf3.fit(X_train, y_train)
```

```
Out[9]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=5, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')
```

```
In [10]: clf3.score(X_test, y_test)
```

```
Out[10]: 0.7777777777777778
```

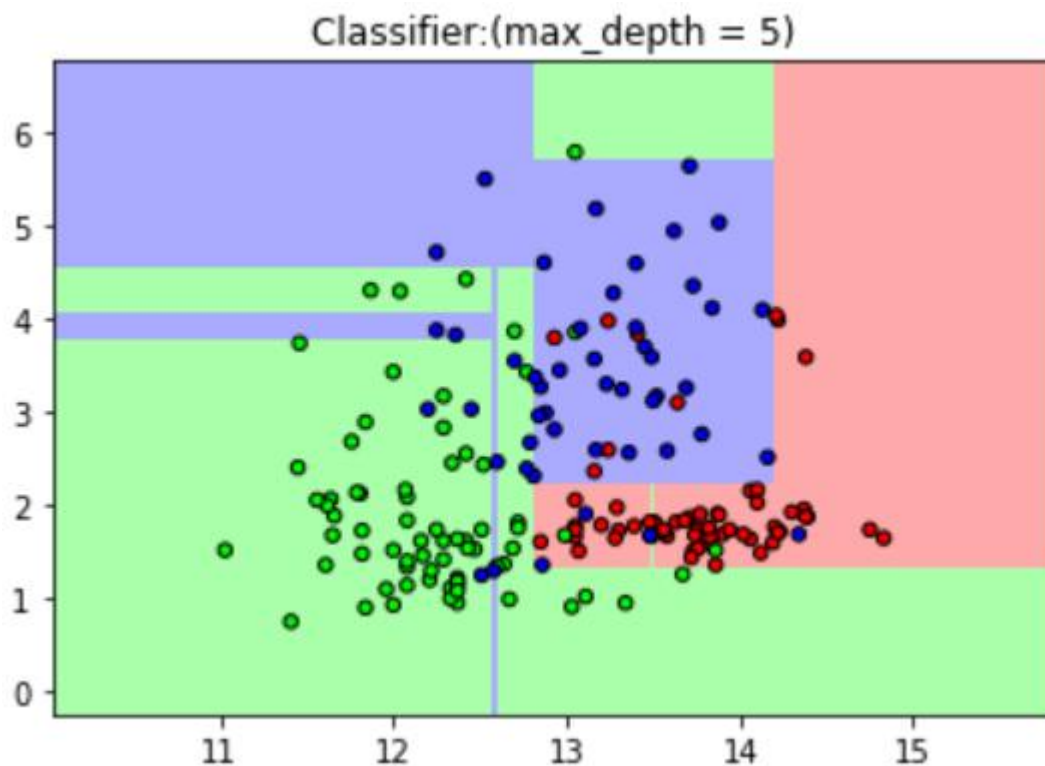
```
In [11]: #定义图像中分区的颜色和散点的颜色
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

#分别用样本的两个特征值创建图像和横轴和纵轴
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                     np.arange(y_min, y_max, .02))
Z = clf3.predict(np.c_[xx.ravel(), yy.ravel()])

#给每个分类中的样本分配不同的颜色
Z = Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

#用散点把样本表示出来
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolor='k', s=20)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Classifier: (max_depth = 5)")

plt.show()
```

结果分析：可以看出，当调整最大深度max_depth=5时分类器的分类性能进一步得到提升，它将更多的数据点放入正确的分类之中。

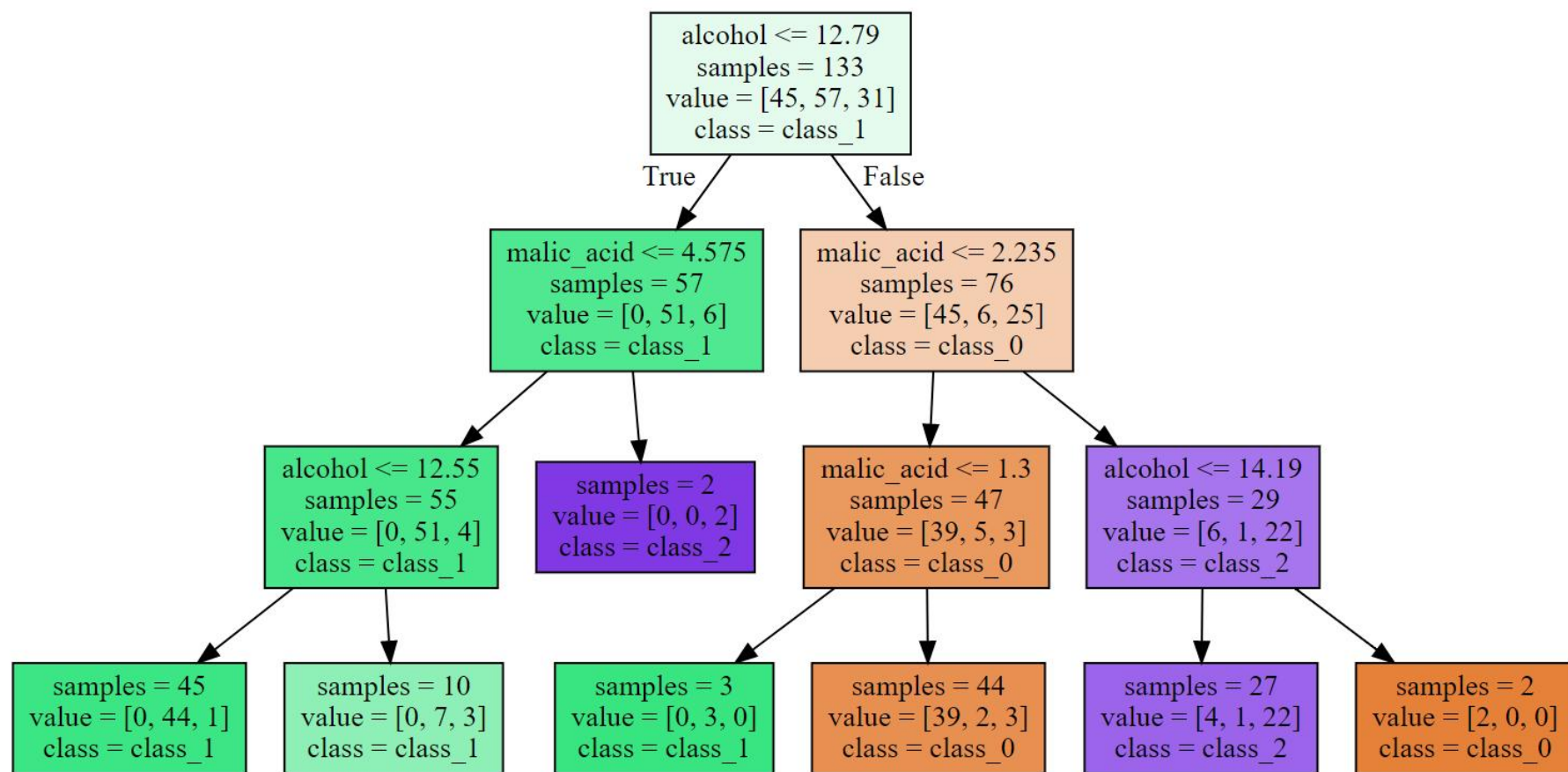
使用graphviz库显示决策树工作过程(非必须)

```
In [12]: #导入graphviz库
import graphviz
#导入决策树中输出graphviz的接口
from sklearn.tree import export_graphviz
```


```
In [13]: #选择最大深度为3的分类模型
export_graphviz(clf2, out_file="wine.dot", class_names=wine.target_names,
feature_names=wine.feature_names[:2], impurity=False, filled=True)
```

```
In [15]: #打开一个dot文件
with open("wine.dot") as f:
    dot_graph = f.read()
#显示dot文件中的图形
graphviz.Source(dot_graph)
```

Out[15]:



上图清晰地展示了决策树是如何进行预测的。先从决策树根结点开始看，第一个条件是**酒精含量**小于或等于12.79，samples=133指在根结点上，有133个样本。value=[45,57,31]是指有45个样本属于class_0，57个样本属于属于class_1，其余31个样本属于class_2。接下来跟着树枝一起前进，在酒精含量小于或等于12.79这个条件为True的情况下，决策树判断分类为class_1，若为False，则判断为class_0；这样到下一层结点，判断为class_1的样本有57个，而判断为class_0的样本有76个；而下一层结点则是对酒的**苹果酸含量**进行判断，进一步对样本进行分类，左边class_1的分支的判断条件是苹果酸含量小于或等于4.575，若为True，则再判断酒精含量是否小于或等于12.55，若为False，则将两个样本归入class_2。依此类推，直到样本全部归入3个分类之中。

 Anaconda Prompt

```
(base) C:\Users\86131>activate tf2
```

```
(tf2) C:\Users\86131>pip install graphviz
```

```
Collecting graphviz
```

```
  Downloading graphviz-0.16-py2.py3-none-any.whl (19 kB)
```

```
Installing collected packages: graphviz
```

```
Successfully installed graphviz-0.16
```

```
(tf2) C:\Users\86131>
```

本章小结:

1. 分类决策树模型是表示基于特征对实例进行分类的树形结构。决策树可以转换成一个if-then规则的集合，也可以看作是定义在特征空间划分上的类的条件概率分布。
2. 决策树学习旨在构建一个与训练数据拟合很好，并且复杂度小的决策树。因为从可能的决策树中直接选取最优决策树是NP完全问题。现实中采用启发式方法学习次优的决策树。

决策树学习算法包括3部分：特征选择、树的生成和树的剪枝。常用算法有ID3、C4.5和CART。本章学习了ID3算法和CART算法。

3. 特征选择的目的在于选取对训练数据能够分类的特征。特征选择的关键是其准则。本章学习的两个准则如下：

- (1) 样本集合D对特征A的信息增益(ID3)

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

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

$$H(D | A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i)$$

其中， $H(D)$ 是数据集 D 的熵， $H(D_i)$ 是数据集 D_i 的熵， $H(D|A)$ 是数据集 D 对特征 A 的条件熵。 D_i 是 D 中特征 A 取第 i 个样本子集， C_k 是 D 中属于第 k 类的样本子集。 n 是特征 A 取值的个数， K 是类的个数。

(2) 样本集合 D 的基尼指数 (CART)

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

特征 A 条件下集合 D 的基尼指数：

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

4. **决策树的生成。**通常使用信息增益最大(ID3)或基尼指数最小(CART)作为特征选择的准则。决策树的生成往往通过计算信息增益等指标，从根结点开始，递归地产生决策树。这相当于用信息增益或其他准则不断地选取局部最优的特征，或将训练集分割为能够基本正确分类的子集。

5. ID3、CART算法比较

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
ID3	分类	多叉树	信息增益	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益比	支持	支持	支持
CART	分类回归	二叉树	基尼指数	支持	支持	支持

本章主要参考文献:

[01] 李航. 统计学习方法(第2版): 67-89, 清华版, 2019

[02] 周志华. 机器学习: 73-95, 清华版, 2016

课外作业7(决策树):

1. 如果训练集有100 万个实例，对叶节点样本数和数的深度均不加约束，则训练一个特征均为二值变量、平衡的二叉决策树，树的大致深度是多少？
2. 考虑下表二分类问题的数据集。

样本序号	A	B	类别
1	T	F	+
2	T	T	+
3	T	T	+
4	T	F	-
5	T	T	+
6	F	F	-
7	F	F	-
8	F	F	-
9	T	T	-
10	T	F	-

- (1)分别计算按照属性A和属性B划分时的信息增益。使用信息增益准则的决策树分类算法应该使用哪个属性？
- (2)分别计算按照属性A和属性B划分时的基尼指数。使用基尼指数的决策树分类算法应该使用哪个属性？

作业参考答案：

1. 答：一个包含 m 个叶节点的均衡二叉树的深度等于 $\log_2(m)$ 的四舍五入。通常来说，二元决策树训练到最后大体都是平衡的，如果不加以限制，最后平均每个叶节点一个实例。因此，如果训练集包含一百万个实例，那么决策树深度约等于 $\log_2(10^6)$ 。因为 $10^3 = 2^{10}$ ，于是 $10^6 = 2^{20}$ ， $\log_2(10^6) = \log_2(2^{20}) = 20$ 。实际上可能会更深一些，因为决策树通常不可能完美平衡。

作业参考答案：

2. (1) Solution:

The contingency tables after splitting on attributes A and B are:

	$A = T$	$A = F$		$B = T$	$B = F$
+	4	0	+	3	1
-	3	3	-	1	5

The overall entropy before splitting is:

$$E_{orig} = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.9710$$

The information gain after splitting on A is:

$$\begin{aligned} E_{A=T} &= -\frac{4}{7} \log \frac{4}{7} - \frac{3}{7} \log \frac{3}{7} = 0.9852 \\ E_{A=F} &= -\frac{3}{3} \log \frac{3}{3} - \frac{0}{3} \log \frac{0}{3} = 0 \\ \Delta &= E_{orig} - 7/10 E_{A=T} - 3/10 E_{A=F} = 0.2813 \end{aligned}$$

The information gain after splitting on B is:

$$\begin{aligned} E_{B=T} &= -\frac{3}{4} \log \frac{3}{4} - \frac{1}{4} \log \frac{1}{4} = 0.8113 \\ E_{B=F} &= -\frac{1}{6} \log \frac{1}{6} - \frac{5}{6} \log \frac{5}{6} = 0.6500 \\ \Delta &= E_{orig} - 4/10 E_{B=T} - 6/10 E_{B=F} = 0.2565 \end{aligned}$$

Therefore, attribute A will be chosen to split the node.

作业参考答案：

2. (2) Solution:

The overall gini before splitting is:

$$G_{orig} = 1 - 0.4^2 - 0.6^2 = 0.48$$

The gain in gini after splitting on A is:

$$G_{A=T} = 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 0.4898$$

$$G_{A=F} = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0$$

$$\Delta = G_{orig} - 7/10G_{A=T} - 3/10G_{A=F} = 0.1371$$

The gain in gini after splitting on B is:

$$G_{B=T} = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.3750$$

$$G_{B=F} = 1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = 0.2778$$

$$\Delta = G_{orig} - 4/10G_{B=T} - 6/10G_{B=F} = 0.1633$$

Therefore, attribute B will be chosen to split the node.

作业参考答案：

3. CART算法按照基尼不纯度 (基尼指数) 进行结点划分，通常来说，子节点的基尼不纯度是高于还是低于其父节点？有没有可能，某个子节点的基尼不纯度高于其父结点的？给出算例说明你的结论。（*：选做）

答：一个节点的基尼不纯度通常比其父节点低，但是也存在一些特殊的情况，某个子节点的基尼不纯度大于其父结点。比如，二分类问题中，某个父结点有5个样本，在类别上分布为：1,2,1,1,1，基尼数为 $1-(1/5)^2-(4/5)^2=0.32$ 。按照基尼不纯度分为两个子结点，左节点样本子集的分类标记为1,2，右节点样本子集的分类标记为1,1,1。显然左结点的基尼数为0.5，大于父结点。但是两个子结点的基尼数的加权和（0.2），仍然低于父结点，因此是一个合理的划分。

End of this lecture.
Thanks !