第 3 章 决策树分类

分类(classification)是人类认识世界的一种重要方法,人类对事物的认识大多是通过分门别类进行的。分类知识反映同类事物共同性质的特征型知识和不同事物之间的差异型特征知识。在机器学习领域,分类是从一个已知类别的数据集到一组预先定义的、非交叠类别的映射过程。其中映射关系的生成以及映射关系的应用是分类算法的主要研究内容。这里的映射关系就是常说的分类器(也叫分类模型或分类函数),映射关系的应用就是使用分类器将测试数据集中的数据划分到给定类别中的某一个类别的过程。

分类从历史的特征数据中构造出特定对象的描述模型(分类器),用来对未来数据进行预测分析,属于机器学习中的预测任务。分类技术使用的历史训练样本数据有确定的类标号,所以分类属于机器学习中的有监督学习。分类技术具有非常广泛的应用领域,如医疗诊断、信用卡系统的信用分级、图像模式识别、网络数据分类等。决策树分类是一种经典分类算法,本章介绍决策树分类的基本概念、常见的 3 种决策树算法,以及分类算法的评价。

3.1 基 本 概 念

分类的目的是得到一个分类器(也称为分类模型),通过得到的分类器能把测试集中的测试数据映射到给定类别中的某一个类别,实现对该数据的预测性描述。分类可用于提取描述重要数据类的模型或预测未来的数据趋势。

3.1.1 什么是分类

对于餐饮企业而言,需要分析数据,如搞清楚不同时节菜品历史的销售情况,分析不同因素下顾客的增加、流失情况等。数据分析的其中一项任务就是分类。分类是从历史的特征数据中构造出特定对象的描述模型或分类器,用来对未来进行预测。分类方法用于预测数据对象的离散类别,如顾客对菜品种类 A、B、C 的喜好,贷款申请数据的“安全”或“危险”。这些类别可以用离散值表示,其中值之间没有次序。例如,可以使用值 1、2、3 表示上面的菜品种类 A、B、C,其中这组菜品种类之间并不存在蕴含的序。下面介绍分类的相关概念。

1. 对象、属性

数据集由数据对象组成。数据集中的数据对象代表一个实体。

属性是一个数据字段,表示数据对象的一个特征。每个数据对象都由若干属性组成。

2. 分类器

分类的关键是找出一个合适的分类函数或分类模型。分类的过程是依据已知的样本数据构造一个分类函数(也叫分类模型或者分类器)。该函数或模型能够把数据对象映射到某一个给定的类别中,从而判别数据对象的类别。

3. 训练集

分类的训练集也称为样本数据，是构造分类器的基础。训练集由数据对象组成，每个对象的所属类别（类标号）已知。在构造分类器时，需要输入一定量的训练集。选取的训练集是否合适，直接影响分类器的性能。

4. 测试集

与训练集一样，测试集也由类别属性已知的数据对象组成。测试集用来测试基于训练集构造的分类器的性能。在分类器产生后，由分类器判定测试集对象的所属类别，将判定的所属类别与测试集已知的所属类别进行比较，得出分类器的正确率等一系列评价性能。

定义：给定一个数据集 $D=\{t_1,t_2,\cdots,t_n\}$ 和一组类 $C=\{C_1,C_2,\cdots,C_m\}$，分类问题是确定一个映射 f：$D\rightarrow C$，使得每个数据对象 t_i 被分配到某一个类中。一个类 C_j 包含映射到该类中的所有数据对象，C 构成数据集 D 的一个划分，即 $C_j=\{t_i \mid f(t_i)=C_j, 1\leqslant i\leqslant n, t_i\in D\}$，并且 $C_j\bigcap C_i=\varnothing$。

分类问题的输入数据（训练集）是由一条条数据对象记录组成的。每条记录包含若干个属性（attribute），组成一个特征向量。训练集的每条记录有一个特定的类别属性（类标号）。该类标号是系统的输入，通常是以往的历史经验数据。分析输入数据，通过在训练集中的数据表现出来的特性，为每个类找到一种准确的描述或者模型。基于此类模型描述对未来的测试数据进行分类，即类别预测。

另外，如果上述定义所构造的模型是预测一个连续值或有序值，而不是离散的类标号，则这种数值预测（numeric prediction）模型通常叫预测器（predictor）。通常将离散型的类标号预测叫分类，将连续型的数值预测叫回归分析（regression analysis）。分类和回归分析是预测问题的两种主要类型。例如，销售经理希望预测一位顾客在一次购物期间将花多少钱。该数据分析任务就是数值预测，也叫回归分析。

3.1.2 分类过程

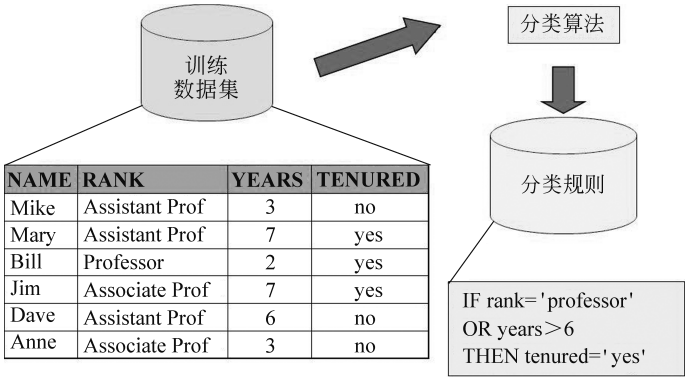
“从例子中学习”（learning from examples）是机器学习中最常用的方法，就是从带有类标号的训练样本中建立分类模型（函数），应用此分类模型对测试样本预测类标号。分类过程主要包含两个步骤：建立模型和应用模型进行分类。

第一步：建立模型。

如图 3.1 所示，通过分析训练数据集构造分类器模型。数据即样本、实例或对象。每个数据属于一个预定义的类，由一个称为类标号的属性确定。为模型建立而被分析的数据形成训练数据集。每个训练样本都有一个特定的类标签与之对应，即对于样本数据 X，其中 x 是训练数据的常规属性，y 是其对应的类标号属性，X 就可以简单表示为二维关系 $X(x,y)$，x 往往包含多个特征值，是多维向量。由于提供了每个训练样本的类标号，因此建模过程是监督学习，即模型的学习过程是在被告知每个训练样本属于哪个类“监督”下进行的。

分类器模型的表示形式有分类规则、决策树或等式、不等式、规则式等形式，这个分类器模型对历史数据分布模型进行了归纳，可以对未来测试数据样本进行分类，也有助于更好地理解数据集的内容或含义。

图 3.1 所示是某校教师情况数据库，常规属性有名字（NAME）、职称（RANK）和工龄（YEARS），类标号属性表示是否获得终身职位（TENURED），学习的分类器以分类规则形式提供。

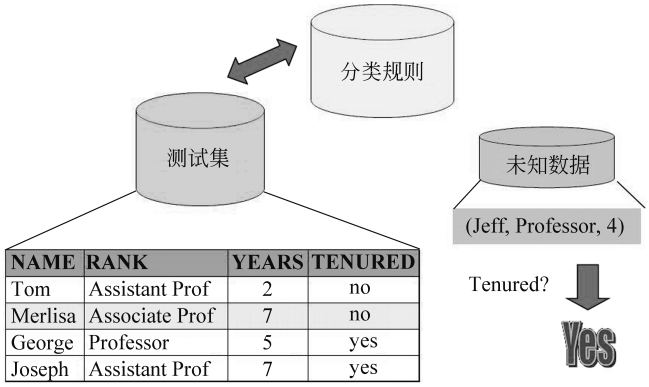


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

图 3.1 建立模型

第二步：应用模型进行分类。

首先根据特定领域的分类器性能要求，对第一步建立的分类器的性能进行科学评估，具体评估方法后续章节中会详细描述。如果该分类器满足研究领域的性能要求，就可以用该分类器对类标号未知的数据或对象进行分类（这种数据在机器学习文献中也称为未知的或先前未见到的数据）。例如，在图 3.2 中通过分析现有教师数据学习得到的分类规则可预测是否给予新的或未来教师终身职位。



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

图 3.2 用模型进行分类

简单来说，分类的两个步骤可以归结为建立模型和应用模型进行分类。模型的建立过程就是使用训练数据进行机器学习的过程，模型的应用过程就是对类标号未知的数据进行分类的过程。

3.1.3 分类器常见的构造方法

分类器常见的构造方法有数理统计方法、决策树方法、神经网络方法和其他方法等。其中数理统计方法有贝叶斯法和非参数法，常见的 K-最近邻（K-Nearest Neighbors，KNN）算法属于非参数方法；神经网络方法有误差反向传播（Error Back Propagation，EBP）算法等；其他方法如粗糙集方法、遗传算法等。本章重点介绍决策树分类方法，后续章节会介绍其他类型的分类器。

3.1.4 决策树分类

1. 决策树分类方法

决策树是机器学习中常用的分类方法。决策树分类方法最后形成的分类模型或者分类器以二叉树或者多叉树的形式表现出来。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，每个分支是根据条件属性取值选取的路径；每个决策结点代表一个问题或决策，通常对应待分类对象的属性；每个叶子结点代表一种使判断终止的可能的分类结果。沿决策树从上到下遍历的过程中，在每个结点都会遇到一个测试，对每个结点上问题的不同的测试输出导致不同的分支，最后会到达一个叶子结点，得到所属类别。人们可以通过决策树直观、准确地得到分类规则，并对未知数据做出客观、准确的分类判断。

决策树算法是基于信息论发展起来的，自 20 世纪 60 年代以来，在分类、预测、规则提取等领域有着广泛的应用。而自从 ID3 算法提出以后，在机器学习、知识发现等领域得到进一步的应用和巨大的发展。经过多年的发展，目前常用的决策树算法有 ID3、C4.5、CART、SLIQ、CHAID 等。决策树归纳的设计问题如下：

（1）如何分裂训练记录？

- 怎样为不同类型的属性指定测试条件？
- 怎样评估每种测试条件？

（2）如何停止分裂？

2. 相关定义

决策树分类算法是以信息论为基础，以信息熵和信息增益度为衡量标准，从而实现对数据的归纳分类。下面是一些信息论的基本概念。

定义 1：熵是指系统存在的一个状态函数，泛指某些物质系统状态的一种度量。信息熵在信息论中称为平均信息量，是对被传送信息进行度量时所采用的一种平均值。

定义 2：若存在 n 个相同概率的消息，则每个消息的概率 p 是 $1/n$，一个消息传递的信息量为 $-\log_2(p)$。p 代表信息发生的可能性，发生的可能性越大，概率越大，信息量越少，通常将这种可能性称为不确定性，即越有可能且越能确定，则信息量越少。

定义 3：若集合 D 有 n 个类别，各类对象的概率分布为 $P=(p_1,p_2,\cdots,p_n)$，则由该集合 D 传递的信息量称为该概率分布 P 的集合信息熵，记为

$$\mathrm{Info}(D)=-\sum_{i=1}^{n}(p_i\log_2 p_i)$$

定义 4：若一个记录集合 D 根据非类别属性 X 的值将其分成集合 $d_1,d_2,\cdots,d_n$，假设对于其中的集合 d_j 根据类别属性的值被分成互相独立的类 $c_{1j},c_{2j},\cdots,c_{ij}$，则其概率分布为 $P=(|c_{1j}|/|d_j|,|c_{2j}|/|d_j|,\cdots,|c_{ij}|/|d_j|)$，则集合 d_j 的信息熵为

$$\mathrm{Info}(d_j)=-\sum_{j=1}^{i}((|c_{ij}|/|d_j|)\log_2(|c_{ij}|/|d_j|))$$

定义 5：若一个记录集合 D 先根据非类别属性 X 的值将 D 分成集合 $d_1,d_2,\cdots,d_n$，则确定 D 中一个元素类别的信息量，可通过确定 d_i 的加权平均值得到，即 $\mathrm{Info}(d_i)$ 的加权平均值为

$$\mathrm{Info}(X,D)=\sum_{j=1}^{n}((|d_j|/|D|)\mathrm{Info}(d_j))$$

定义6：信息增益是两个信息量之间的差值，是度量样本集合纯度的指标，其中一个信息量是需确定 D 中一个元素所属分类需要的信息量，另一个信息量是在已得到的非类别属性 X 的值后确定 D 中一个元素所属分类需要的信息量，集合 D 根据非类别属性 X 进行划分的信息增益为

$$\mathrm{Gain}(X,D)=\mathrm{Info}(D)-\mathrm{Info}(X,D)$$

定义7：信息增益率是一种增益率，是对信息增益进行改进，考虑分裂后每个子结点的样本数量的纯度，集合 D 根据非类别属性 X 进行划分的信息增益率的计算方法如下。

$$\mathrm{Gain-Ratio}(X,D)=\mathrm{Gain}(X,D)/\mathrm{Info}(X,D)$$

定义8：基尼(Gini)指数是一种不等性度量，通常用来度量收入不平衡，但是它可以用来度量任何不均匀分布。Gini指数是一个0～1的数。其中0对应完全相等(其中每个人都具有相同的收入)，而1对应完全不相等(其中一个人具有所有收入，而其他人收入都为零)。

在分类中，Gini指数度量数据集 D 的不纯度，定义为

$$\mathrm{Gini}(D)=1-\sum_{i=1}^{m}p_i^2$$

其中 p_i 是 D 中元组属于 C_i 类的概率，m 表示类别数。Gini指数越小，则数据集 D 的纯度越高。Gini指数考虑每个属性的二元划分。

(1) 首先考虑属性 A 是离散值属性的情况，其中 A 具有 v 个不同值出现在 D 中。如果 A 具有 v 个可能的值，则存在 2^v 个可能的子集。例如，如果 income 具有3个可能的值{low, medium, high}，则可能的子集具有8个。不考虑幂集({ low, medium, high})和空集({ })，因为从概念上讲，它不代表任何分裂。因此，基于 A 的二元划分，存在 2^v-2 种形成数据集 D 的两个分区的可能方法。

当考虑二元划分裂时，计算每个结果分区的不纯度的加权和。例如，如果 A 的二元划分将 D 划分成 D_1 和 D_2，则给定该划分，D 的Gini指数为

$$\mathrm{Gini}_A(D)=\frac{|D_1|}{|D|}\mathrm{Gini}(D_1)+\frac{|D_2|}{|D|}\mathrm{Gini}(D_2)$$

选择该属性产生最小Gini指数的子集作为它的分裂子集。

(2) 对于连续值属性，其策略类似于上面介绍的信息增益所使用的策略。

离散或连续值属性 A 的二元划分导致的不纯度降低为

$$\Delta\mathrm{Gini}(A)=\mathrm{Gini}(D)-\mathrm{Gini}_A(D)$$

最大化不纯度降低(或等价地，具有最小Gini指数)属性选为分裂属性。该属性和它的分裂子集(对于离散值的分裂属性)或分裂点(对于连续值的分裂属性)一起形成分裂准则。

3.2 CART 算 法

3.2.1 CART 算法介绍

CART算法又叫分类回归树(Classification And Regression Tree，CART)模型，由Breiman等在1984年提出，是应用广泛的决策树学习方法。它采用与传统统计学完全不同的方式构建预测准则，以二叉树的形式给出，易于理解、使用和解释。由CART模型构建的预测树在很多情况下比常用的统计方法构建的代数学预测准则更加准确，数据越复杂、变量

越多，算法的优越性就越显著。

3.2.2 CART 算法原理

CART 算法假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。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，也就是在输入给定的条件下输出的条件概率分布。

在决策树 CART 算法中，用 Gini 指数衡量数据的不纯度或者不确定性，同时用 Gini 指数决定类别变量的最优二分值的切分问题。

在分类问题中，假设样本集合 D_1 有 M 个类，样本点属于第 k 类的概率为 P_k，则概率分布的 Gini 指数的定义为

$$\mathrm{Gini}(D_1)=\sum_{k=1}^{M}p_k(1-p_k)=1-\sum_{k=1}^{M}p_k^2$$

如果样本集合 D 根据某个属性特征 A 被分割为 D_1、D_2 两部分，那么在属性特征 A 的条件下，集合 D 的 Gini 指数的定义为

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

Gini 指数 Gini(D,A)表示属性特征 A 不同分组的数据集 D 的不确定性。Gini 指数值越大，样本集合的不确定性就越大，这一点与熵的概念比较类似。可以通过 Gini 指数确定某个特征的最优切分点（即只需要确保切分后某点的 Gini 指数值最小），这就是决策树 CART 算法中类别变量切分的关键所在。

CART 算法的步骤如下。

步骤 1：设结点的训练数据集为 D，计算现有属性特征对该数据集的 Gini 指数。此时，对每个属性特征 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，计算 $A=a$ 时的 Gini 指数。

步骤 2：在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择 Gini 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。

步骤 3：对两个子结点递归地调用步骤 1～2，直至满足停止条件。

步骤 4：生成 CART 决策树。

CART 算法停止计算的条件是结点中的样本个数小于预定阈值，或训练集的 Gini 指数小于预定阈值（样本基本属于同一类），或者没有更多的特征。

3.2.3 CART 算法实例

【例 3.1】 基于 CART 算法的原理分析得出：可以通过 Gini 指数确定某个特征的最优切分点，即只需确保切分后某点的 Gini 指数最小，这就是决策树 CART 算法中类别变量切分的关键。例如，在银行的贷款业务中，可以通过“是否有房”“婚姻状况”“年收入”等因素分析客户是否拖欠贷款，如表 3.1 所示。下面通过 CART 算法对贷款业务的客户数据影响因素进行分析，判断客户是否拖欠贷款。

表 3.1 贷款业务的客户数据集

序号	是否有房	婚姻状况	年 收 入	是否拖欠贷款
1	yes	single	125k	no
2	no	married	100k	no
3	no	single	70k	no
4	yes	married	120k	no
5	no	divorced	95k	yes
6	no	married	60k	no
7	yes	divorced	220k	no
8	no	single	85k	yes
9	no	married	75k	no
10	no	single	90k	yes

首先对数据集非类标号属性{是否有房,婚姻状况,年收入}分别计算它们的 Gini 指数,取 Gini 指数最小的属性作为决策树的根结点属性。

1. "是否有房"属性的 Gini 指数

Gini(有房)$=1-(0/3)^2-(3/3)^2=0$

Gini(没房)$=1-(3/7)^2-(4/7)^2=0.4898$

Gini(D,是否有房)$=7/10\times0.4898+3/10\times0=0.34286$

2. "婚姻状况"属性的 Gini 指数

若按婚姻状况属性划分,属性婚姻状况有 3 个可能的取值:married、single、divorced,分别计算划分后的 Gini 指数:

{married} | {single,divorced}

{single} | {married,divorced}

{divorced} | {single,married}

当分组为{married} | {single,divorced}时:

Gini(婚姻状况)$=4/10\times0+6/10\times[1-(3/6)^2-(3/6)^2]=0.3$

当分组为{single} | {married,divorced}时:

Gini(婚姻状况)$=4/10\times0.5+6/10\times[1-(1/6)^2-(5/6)^2]=0.367$

当分组为{divorced} | {single,married}时:

Gini(婚姻状况)$=2/10\times0.5+8/10\times[1-(2/8)^2-(6/8)^2]=0.4$

对比计算结果,根据婚姻状况属性划分根结点时取 Gini 指数最小的分组作为划分结果,也就是{married} | {single,divorced} Gini(婚姻状况)为 0.3。

3. "年收入"的 Gini 指数

年收入属性为数值型属性,首先需要对数据按升序排列,然后从小到大依次用相邻值的中间值作为分界点将样本划分为两组。例如,当面对年收入为 60 和 70 这两个值时,我们算得其中间值为 65。倘若以中间值 65 作为分界点,左子树表示年收入小于 65 的样本,右子树表示年收入大于或等于 65 的样本,于是 Gini 指数为

$$\text{Gini}(\text{年收入}) = 1/10\times 0 + 9/10\times[1-(6/9)^2-(3/9)^2] = 0.4$$

同理可得其他分组的 Gini 指数，这里不再逐一给出计算过程，仅列出如下结果（最终取其中使 Gini 系数最小化的那个二分准则作为构建二叉树的准则），如表 3.2 所示。

表 3.2　CART 对“年收入”属性候选划分结点的计算

年收入	60	70	75	85	90	95	100	120	125	220
相邻中值点	65	72.5	80	87.5	92.5	97.5	110	122.5	172.5	
Gini 指数	0.4	0.375	0.343	0.417	0.4	0.3	0.343	0.375	0.4	

Gini 指数最小的是以 97.5 为分界点划分，Gini(年收入) = 0.3。

根据计算可知，3 个属性划分根结点的 Gini 指数最小的有两个：婚姻状况和年收入属性，均为 0.3。此时，选取首先出现的属性婚姻状况作为第一次划分。

接下来，采用同样的方法，分别计算剩下的属性——是否有房和年收入，其中根结点的 Gini 指数与前面的计算过程类似，对于“是否有房”属性，可得：

$$\text{Gini}(\text{是否有房}) = 4/6\times[1-(3/4)^2-(1/4)^2] + 2/6\times 0 = 0.25$$

对于“年收入”属性，相邻中间值为 77.5 时 Gini 指数最小，Gini(年收入) = 0.25，结果如表 3.3 所示。

表 3.3　CART 对“年收入”属性候选划分结点的计算

年收入	70k	85k	90k	95k
相邻中值点	77.5	87.5	92.5	
Gini 指数	0.25	0.25	0.33	

最后通过 CART 算法对贷款业务的客户数据集进行分析，可得如图 3.3 所示的二叉树。

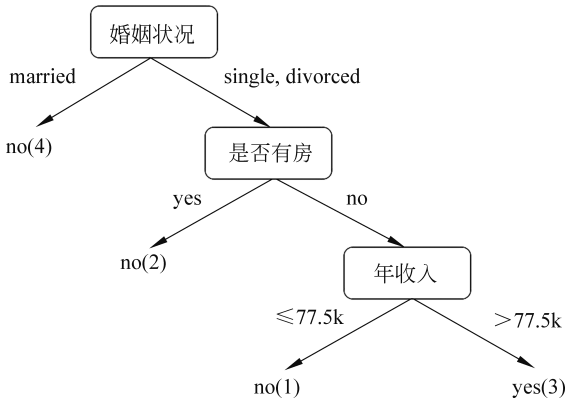


图 3.3　CART 算法实现贷款业务客户数据集的决策树

3.2.4　CART 算法 Python 实现

针对上节用 CART 算法实现贷款业务的客户数据集实例，首先对贷款业务的客户数据集进行初始化，然后随机选取 20% 客户数据作为测试集，80% 作为训练集，再使用 sklearn

库函数训练模型，最后对分类结果进行预测。Python 实现过程为：

```
#数据集定义
if __name__ == '__main__':
    #读取数据
    raw_data = pd.read_excel('./CART_贷款预测数据集.xlsx', header=0)
    data = raw_data.values
    features = data[::,:2]
    label = data[::,3]
    #随机选取 20%数据作为测试集，剩余为训练集
    train_features, test_features, train_labels, test_labels = train_test_
split(features, label, test_size=0.2, random_state=0)
    #采用 CART 算法训练，并预测
    clf = DecisionTreeClassifier(criterion='gini')
    clf.fit(train_features,train_labels)
    test_predict = clf.predict(test_features)
    #准确性评分
    score = accuracy_score(test_labels, test_predict)
```

结合 CART 算法的核心思想是通过 Gini 指数确定某个特征的最优切分点，在该实例中用贷款业务的客户数据集估计 CART 算法分类的准确率。

3.2.5　CART 算法的优缺点

通过对用 CART 算法实现贷款业务的客户数据集实例的过程分析，以及用 Python 代码实现其过程，分析 CART 算法的优缺点。

CART 算法的优点如下。

(1) CART 算法采用了简化的二叉树模型，同时特征选择采用 Gini 指数简化计算。

(2) CART 树最大的好处是可以做回归模型。

CART 算法的缺点如下。

(1) CART 在做特征选择的时候都是选择最优的一个特征做分类决策，但是大多数情况分类决策不应该由某一个特征决定，而是由一组特征决定。根据一组特征构造的决策树往往更加准确。

(2) 如果训练样本数据发生改动，就可能导致树结构发生剧烈改变。

3.3　ID3 算 法

3.3.1　ID3 算法介绍

ID3 算法最早是由罗斯昆(Ross Quinlan)于 1975 年在悉尼大学提出的一种分类预测算法。ID3 算法是一种贪心算法，起源于概念学习系统(Concept Learning System，CLS)，以信息增益作为选取分裂属性的标准建立决策树，即在每个结点选取尚未被用来划分的具有最高信息增益的属性作为划分标准。

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集合的测试属性。对被选取的测试属性创建一个结点，并以该结点的属性作为标记，对该属性的每个值

创建一个分支以此划分样本。

3.3.2 ID3 算法原理

ID3 算法的核心思想是通过计算属性的信息增益选择决策树各级结点上的分类属性，使得在每个非叶子结点都可获得被测样本的最大类别信息。

ID3 算法的基本思路是：计算所有属性的信息增益，选择其中最大信息增益的属性作为分裂属性产生决策树结点，然后根据该属性的不同取值建立相同数量的分支，再对各分支递归调用 ID3 算法建立分支，直到子集中仅包括同一类别或没有可分裂的属性为止。由此可以得到一棵决策树，对样本进行分析预测。

ID3 算法流程图如图 3.4 所示。

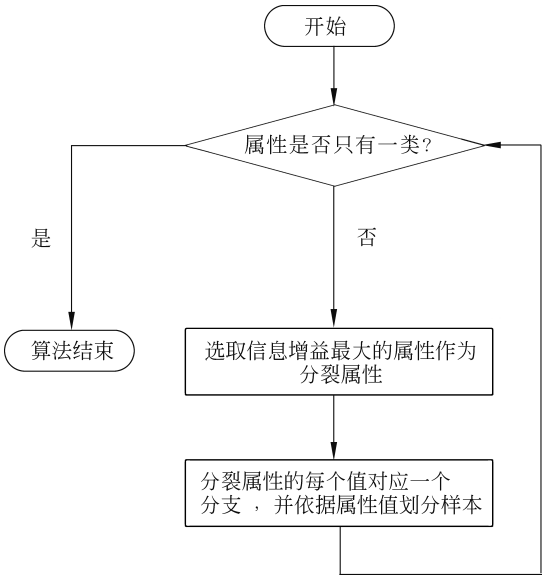


图 3.4 ID3 算法流程图

训练集按照属性划分信息增益的计算步骤如下。

(1) 计算给定的训练集的信息熵 entropy。

设训练集为 S，类标号属性将训练集分为 n 个子集：$S_1, S_2, \cdots, S_n$。设 d 为训练集 S 中所有样本的总数，d_i 表示子集 S_i 中的个数，概率为

$$P_i = \frac{d_i}{d}$$

给定训练集的信息熵为

$$\text{entropy}(S) = -\sum_{i=1}^{n}(P_i \log 2 P_i)$$

(2) 计算训练集中的某一属性 A，若属性 A 有 m 个取值，计算每个取值 a_j 的信息熵 $\text{entropy}(a_j)$。

设 d_j 为 $A = a_j$ 的样本总数，d_{ij} 表示当 $A = a_j$ 时对应不同类标号属性的取值个数，则

$$\text{entropy}(a_j) = -\sum_{i=1}^{n}\frac{d_{ij}}{d_j}\log\left(\frac{d_{ij}}{d_j}\right)$$

(3) 计算属性 A 的划分信息熵 entropy(A)：

$$\text{entropy}(S,A)=\sum_{j=1}^{m}\frac{d_j}{d}\text{entropy}(a_j)$$

(4) 计算属性 A 的信息增益 Gain(A)：

$$\text{Gain}(A)=\text{entropy}(S)-\text{entropy}(S,A)$$

3.3.3　ID3 算法实例

【例 3.2】 在分析 ID3 算法的实现过程后，本节将通过 ID3 算法对表 3.4 所示的游客外出游玩天气数据集进行分析构造决策树，评估游客决定外出游玩时与哪些属性相关。

表 3.4 是一张二维表，共有 outlook、temperature、humidity、windy、play 5 个属性，要分析的就是前 4 个属性对属性 play 的影响。

表 3.4　游客外出游玩天气数据集

No.	1：outlook Nominal	2：temperature Nominal	3：humidity Nominal	4：windy Nominal	5：play Nominal
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

通过对表 3.4 进行整理，针对不同的属性对游客是否可以外出游玩进行评估，得到如表 3.5 所示的天气数据集属性分析表。

表 3.5　天气数据集属性分析表

属　　性	no	yes	no	yes	no	yes
play	5	9				
outlook	3	2	0	4	2	3
	sunny　5		overcast　4		rainy　5	

续表

属　性	no	yes	no	yes	no	yes
temperature	2	2	2	4	1	3
	hot　4		mild　6		cool　4	
humidity	4	3	1	6		
	high　7		normal　7			
windy	3	3	2	6		
	true　6		false　8			

(1) 计算属性 play 的信息熵：

$$entropy=-(5/14)\log_2(5/14)-(9/14)\log_2(9/14)=0.941$$

(2) 计算其余各个属性的每个取值的信息熵：

$$entropy(outlook=sunny)=-(3/5)\log_2(3/5)-(2/5)\log_2(2/5)=0.971$$

$$entropy(outlook=overcast)=-(4/4)\log_2(4/4)=0$$

$$entropy(outlook=rainy)=-(3/5)\log_2(3/5)-(2/5)\log_2(2/5)=0.971$$

$$entropy(temperature=hot)=-(2/4)\log_2(2/4)-(2/4)\log_2(2/4)=1$$

$$entropy(temperature=mild)=-(4/6)\log_2(4/6)-(2/6)\log_2(2/6)=0.918$$

$$entropy(temperature=cool)=-(3/4)\log_2(3/4)-(1/4)\log_2(1/4)=0.811$$

$$entropy(humidity=high)=-(3/7)\log_2(3/7)-(4/7)\log_2(4/7)=0.985$$

$$entropy(humidity=normal)=-(1/7)\log_2(1/7)-(6/7)\log_2(6/7)=0.592$$

$$entropy(windy=true)=-(3/6)\log_2(3/6)-(3/6)\log_2(3/6)=1$$

$$entropy(windy=false)=-(6/8)\log_2(6/8)-(2/8)\log_2(2/8)=0.811$$

(3) 计算各属性的信息熵：

$$entropy(outlook)=5/14*0.971+4/14*0+5/14*0.971=0.693$$

$$entropy(temperature)=4/14*1+6/14*0.918+4/14*0.811=0.911$$

$$entropy(humidity)=7/14*0.985+7/14*0.592=0.789$$

$$entropy(windy)=6/14*1+8/14*0.811=0.892$$

(4) 计算各属性的信息增益：

$$Gain(outlook)=0.941-0.693=0.248$$

$$Gain(temperature)=0.941-0.911=0.03$$

$$Gain(humidity)=0.941-0.789=0.152$$

$$Gain(windy)=0.941-0.892=0.049$$

由此可知，应选择属性 outlook 作为根结点，得出如图 3.5 所示的树状图。

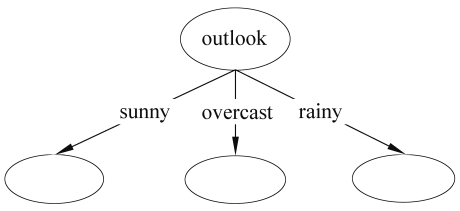


图 3.5　第一步根结点分析结果

(5) 分析可知 overcast 对应的属性 play 值全为 yes，无须进行分类，现先对 sunny 这一分支进行分析，如表 3.6 所示。

表 3.6　sunny 分支属性表

temperature	humidity	windy	play
hot	high	false	no
hot	high	true	no
mild	high	false	no
cool	normal	false	yes
mild	normal	true	yes

① 计算属性 play 的信息熵：

$entropy=-(3/5)\log_2(3/5)-(2/5)\log_2(2/5)=0.971$

② 计算其余各属性的每个取值的信息熵：

$entropy(temperature=hot)=-(2/2)\log_2(2/2)=0$

$entropy(temperature=mild)=-(1/2)\log_2(1/2)-(1/2)\log_2(1/2)=1$

$entropy(temperature=cool)=-(1/1)\log_2(1/1)=0$

$entropy(humidity=high)=-(3/3)\log_2(3/3)=0$

$entropy(humidity=normal)=-(2/2)\log_2(2/2)=0$

$entropy(windy=true)=-(1/2)\log_2(1/2)-(1/2)\log_2(1/2)=1$

$entropy(windy=false)=-(2/3)\log_2(2/3)-(1/3)\log_2(1/3)=0.918$

③ 计算各属性的信息熵：

$entropy(temperature)=2/5*1=0.4$

$entropy(humidity)=0$

$entropy(windy)=2/5*1+3/5*0.918\approx0.951$

④ 计算各属性的信息增益：

$Gain(temperature)=0.971-0.4=0.571$

$Gain(humidity)=0.971-0=0.971$

$Gain(windy)=0.971-0.951=0.02$

由此可知应选择属性 humidity 作为根结点，且无须再进行分类。对属性取值 sunny 分支进行计算得到的树状图如图 3.6 所示。

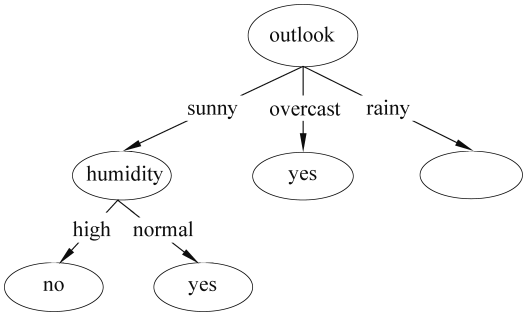


图 3.6　第二步属性 sunny 分析结果

（6）对 rainy 分支进行分析，如表 3.7 所示。

表 3.7　rainy 分支属性表

temperature	windy	play
mild	false	yes
cool	false	yes
cool	true	no
mild	false	yes
mild	true	no

① 计算属性 play 的信息熵：

$entropy=-(3/5)\log_2(3/5)-(2/5)\log_2(2/5)=0.971$

② 计算其余各属性的每个取值的信息熵：

$entropy(temperature=mild)=-(2/3)\log_2(2/3)-(1/3)\log_2(1/3)=0.918$

$entropy(temperature=cool)=(-(1/2)\log_2(1/2))*2=1$

$entropy(windy=true)=-(2/2)\log_2(2/2)=0$

$entropy(windy=false)=-(3/3)\log_2(3/3)=0$

③ 计算各属性的信息熵：

$entropy(temperature)=2/5*1+3/5*0.918=0.951$

$entropy(windy)=0$

④ 计算各属性的信息增益：

$Gain(temperature)=0.971-0.951=0.02$

$Gain(windy)=0.971-0=0.971$

由此可知应选择属性 windy 作为根结点，且无须再进行分类，则得出用 ID3 算法对游客外出游玩数据集分析的最终决策树，如图 3.7 所示。

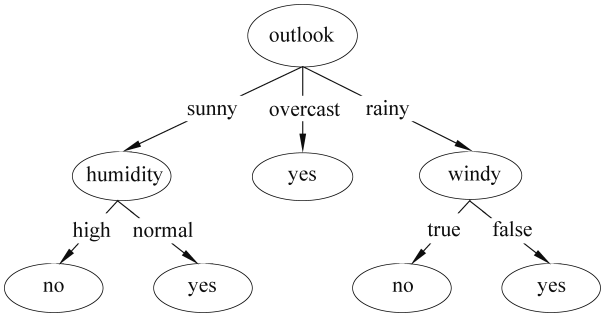


图 3.7　游客外出游玩决策树

3.3.4　ID3 算法 Python 实现

ID3 算法的实现思想：每次从数据集中根据最大信息增益选取一个决策属性特征，将数据进行划分，每次划分都会消耗一个属性特征，使得属性特征越来越少，当所有数据集都是同一类，或者消耗完所有属性特征时，划分结束。

针对3.3.3节的ID3算法实例，对是否外出出游进行预测，首先将天气outlook、温度temperature、湿度humidity等信息通过createDataSet()方法进行初始化，包含4个属性：outlook、temperature、humidity、windy、play。以天气为例，0代表雨天，1代表阴天，2代表晴天。然后通过创建树计算各属性信息增益，最后通过对待分类样本进行分类得到结果，即是否外出。Python实现过程如下。

```
#数据初始化,属性分别为天气、温度、湿度、风力
def createDataSet():
    #outlook: 0 rain   1 overcast   2 sunny
    #tem:     0 cool   1 mild       2 hot
    #hum:     0 normal 1 high
    #windy    0 not    1 medium     2 very
    dataSet = pd.read_excel('./ID3_天气预测数据集', header=0)
    train_data = np.array(dataSet)
    excel_list = train_data.tolist()
    return excel_list

#获取数据集的熵
def calcShannonEnt(dataSet):
    numEntries = len(dataSet)
    labelCounts = {}
    for featVec in dataSet:
        currentLable = featVec[-1]                    #取得最后一列数据
        if currentLable not in labelCounts.keys():    #获取结果
            labelCounts[currentLable] = 0
        labelCounts[currentLable] += 1

    #计算熵
    Ent = 0.0
    for key in labelCounts:
        prob = float(labelCounts[key]) / numEntries
        Ent -= prob * log(prob, 2)
    #print ("信息熵: ", Ent)
    return Ent

#划分数据集
def splitDataSet(dataSet, axis, value):
    retDataSet = []
    for featVec in dataSet:
        #每行中第 axis 个元素和 value 相等(去除第 axis 个数据)
        if featVec[axis] == value:
            reducedFeatVec = featVec[:axis]
            reducedFeatVec.extend(featVec[axis + 1:])
            retDataSet.append(reducedFeatVec)
    return retDataSet                                 #返回分类后的新矩阵

#根据香农熵选择最优的划分方式
#根据某一属性划分后,类标签香农熵越低,效果越好
def chooseBestFeatureToSplit(dataSet):
```

```
    baseEntropy = calcShannonEnt(dataSet)              #计算数据集的香农熵
    numFeatures = len(dataSet[0]) - 1
    bestInfoGain = 0.0                                 #最大信息增益
    bestFeature = 0                                    #最优特征

    for i in range(0, numFeatures):
        #所有子列表(每行)的第 i 个元素,组成一个新列表
        featList = [example[i] for example in dataSet]
        uniqueVals = set(featList)
        newEntorpy = 0.0
        #数据集根据第 i 个属性进行划分,计算划分后数据集的香农熵
        for value in uniqueVals:
            subDataSet = splitDataSet(dataSet, i, value)
            prob = len(subDataSet) / float(len(dataSet))
            newEntorpy += prob * calcShannonEnt(subDataSet)
        #划分后的数据集,香农熵越小越好,即信息增益越大越好
        infoGain = baseEntropy - newEntorpy
        if (infoGain > bestInfoGain):
            bestInfoGain = infoGain
            bestFeature = i
    return bestFeature

#如果数据集已经处理了所有属性,但叶子结点中类标签依然不是唯一的,此时需要决定
#如何定义该叶子结点。这种情况下,采用多数表决方法,对该叶子结点进行分类
def majorityCnt(classList):                            #传入参数:叶子结点中的类标签
    classCount = {}
    for vote in classList:
        if vote not in classCount.keys():
            classCount[vote] = 0
            classCount[vote] += 1
    sortedClassCount = sorted(classCount.iteritems(), key = operator.
itemgetter(1), reverse=True)
    return sortedClassCount[0][0]

#创建树
def createTree(dataSet, labels):
#传入参数:数据集,属性标签(属性标签的作用:在输出结果时,决策树的构建更加清晰)
    classList = [example[-1] for example in dataSet]  #数据集样本的类标签
    if classList.count(classList[0]) == len(classList):
     #如果数据集样本属于同一类,说明该叶子结点划分完毕
        return classList[0]
    if len(dataSet[0]) == 1:
    #如果数据集样本只有一列(该列是类标签),说明所有属性都划分完毕,
    #则根据多数表决方法对该叶子结点进行分类
        return majorityCnt(classList)
    bestFeat = chooseBestFeatureToSplit(dataSet)       #根据香农熵选择最优划分方式
    bestFeatLabel = labels[bestFeat]                   #记录该属性标签
    myTree = {bestFeatLabel: {}}                       #树
    del (labels[bestFeat])                             #在属性标签中删除该属性
    #根据最优属性构建树
```

```
        featValues = [example[bestFeat] for example in dataSet]
        uniqueVals = set(featValues)
        for value in uniqueVals:
            subLabels = labels[:]
            subDataSet = splitDataSet(dataSet, bestFeat, value)
            myTree[bestFeatLabel][value] = createTree(subDataSet, subLabels)
        #print("myTree:", myTree)
        return myTree

    #测试算法:使用决策树,对待分类样本进行分类
    def classify(inputTree, featLabels, testVec):
    #传入参数:决策树,属性标签,待分类样本
        firstStr = list(inputTree.keys())[0]              #树根代表的属性
        secondDict = inputTree[firstStr]
        #树根代表的属性,所在属性标签中的位置,即第几个属性
        featIndex = featLabels.index(firstStr)
        for key in secondDict.keys():
            if testVec[featIndex] == key:
                if type(secondDict[key]).__name__ == 'dict':
                    classLabel = classify(secondDict[key], featLabels, testVec)
                else:
                    classLabel = secondDict[key]
        return classLabel

    if __name__ == '__main__':
        dataSet = createDataSet()
        labels = ['outlook', 'tem', 'hum', 'windy']
        labelsForCreateTree = labels[:]
        Tree = createTree(dataSet, labelsForCreateTree)
        testvec = [2, 2, 1, 0]
        classify(Tree, labels, testvec)
```

3.3.5 ID3的优缺点

通过上述的理论描述和实例分析可知,ID3算法的优缺点总结如下。

ID3算法的优点:

(1) 理论清晰,方法简单,学习能力较强。

(2) 生成的规则容易理解。

(3) 适用于处理大规模的学习问题。

(4) 构建决策树的速度较快,计算时间随着数据的增加线性增加。

(5) ID3算法不存在无解的危险,并且全盘使用训练数据,可得到一棵较为优化的决策树。

ID3算法的缺点:

(1) ID3算法在属性选择时,倾向选择那些拥有多个属性值的属性作为决策属性,而这些属性不一定是最佳决策属性。

(2) 只能处理离散属性,对于连续型的属性,在分类前需要对其进行离散化的处理。

(3) 无法对决策树进行优化,生成的决策树易过拟合。

（4）ID3 算法不能增量地接受训练集，每增加一次实例，就抛弃原有的决策树，重新构造新的决策树，开销很大。

3.4 C4.5 算 法

3.4.1 C4.5 算法介绍

C4.5 算法是在 ID3 算法的基础上进行改进的，因此继承了 ID3 算法的优点，并改进了其缺点，改进的部分有以下几点。

（1）将连续型属性变量进行离散化的方法如下。

- 从原始数据中找到该连续型属性的最大值 max 和最小值 min；
- 设置区间[min，max]中的 N 个等分段点 A_i，其中 i 的取值为 1，2，…，N；
- 有多种分段方式，一般将连续属性分为两类，且从小到大排序，分裂结点取为当前分裂点与下一属性值的中间值，取分裂点的信息增益率最大值作为分裂点；
- 分别把[min，A_i]和[A_i，max]作为该连续型属性变量的两类取值，有 $N-1$ 种划分方式。

（2）使用信息增益率进行分裂属性的选择，克服了 ID3 算法用信息增益选择属性时偏向选择取值多的属性的不足。

（3）在决策树的构造过程中进行剪枝，因为某些具有很少元素的结点可能直接导致构造的决策树过拟合，若忽略这些结点，可能更好。剪枝方法可用来处理过分拟合数据的问题。

3.4.2 C4.5 算法原理

C4.5 算法选择信息增益率最大的属性作为决策属性。C4.5 算法的流程如图 3.8 所示。

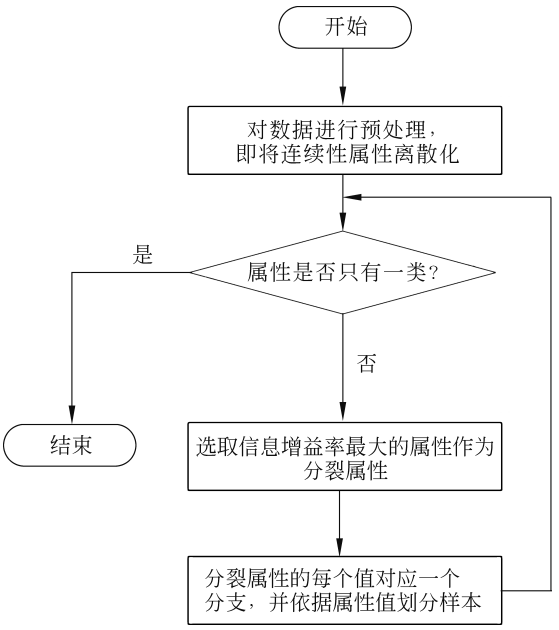


图 3.8 C4.5 算法的流程

C4.5 算法的具体步骤如下。

步骤 1：对数据集进行预处理，将连续型的属性变量进行离散化，处理形成决策树的训练集，若没有连续取值的属性，则忽略此步骤。

步骤 2：计算每个属性的信息增益和信息增益率，并选择信息增益率最大的属性作为当前的属性结点，得到决策树的根结点。

其中信息增益的计算方法和 ID3 算法完全一致。对于取值连续的属性而言，分别计算以分割点对应分类的信息增益率，选择最大信息增益率对应的 A_i 作为该属性分类的分割点，其中 i 的取值为 $1,2,\cdots,N$；

步骤 3：根结点属性每一个可能的取值对应一个子集，对样本子集递归地执行步骤 2 过程，直到划分的每个子集中的观测数据在分类属性上取值都相同，生成决策树。

步骤 4：根据构造的决策树提取分类规则，对新的数据对象进行分类预测。

3.4.3 C4.5 算法实例

【例 3.3】 对如表 3.8 所示的游客外出游玩天气数据集进行分析，通过表中 outlook、temperature、humidity、windy 4 个属性值对属性 play 进行评估，以此判断游客是否适合出游。其中，表 3.8 中的温度属性 temperature 和湿度属性 humidity 是连续值。

表 3.8 游客外出游玩天气数据集

No	outlook Nominal	temperature Numeric	humidity Numeric	windy Nominal	play Nominal
1	sunny	85.0	85.0	false	no
2	sunny	80.0	90.0	true	no
3	overcast	83.0	86.0	false	yes
4	rainy	70.0	96.0	false	yes
5	rainy	68.0	80.0	false	yes
6	rainy	65.0	70.0	true	no
7	overcast	64.0	65.0	true	yes
8	sunny	72.0	95.0	false	no
9	sunny	69.0	70.0	false	yes
10	rainy	75.0	80.0	false	yes
11	sunny	75.0	70.0	true	yes
12	overcast	72.0	90.0	true	yes
13	overcast	81.0	75.0	false	yes
14	rainy	71.0	91.0	true	no

1. 数据预处理

将温度属性 temperature 和湿度属性 humidity 离散化，假设采用的离散化方法是将各个属性值从小到大排序，依次计算其相邻属性值的中值并将其作为分裂点，计算其信息增益，选其最大值。

2. 分析 temperature 属性

（1）找到属性 temperature 的最大值 85 和最小值 64，如表 3.9 所示。

表 3.9　temperature 分裂点

分　裂　点	区　　间	总数(14)	yes(9)	no(5)
65	[64,65]	2	1	1
	(65,85]	12	8	4
69	[64,69]	4	3	1
	(69,85]	10	6	4
70	[64,70]	5	4	1
	(70,85]	9	5	4
71	[64,71]	6	4	2
	(71,85]	8	5	3
72	[64,72]	8	5	3
	(72,85]	6	4	2
75	[64,75]	10	7	3
	(75,85]	4	2	2
81	[64,81]	12	8	4
	(81,85]	2	1	1
84	[64,84]	13	9	4
	(84,85]	1	0	1

在区间[64,85]中取分裂点，根据 C4.5 算法，要将属性值进行排序遍历到每一个值，然后找出信息增益最大的值作为该属性的分裂点。为求得[min,A_i]和[A_i,max]最大的信息增益，则信息熵就要最小，现做统计，如表 3.9 所示：把作为该连续型属性变量的两类取值分别命名为 no1 和 no2。

其中，分裂点为 65 与 81 的取值分布相同，分裂点为 71 与 72 的取值分布相同，可简化计算。

（2）计算属性 play 的信息熵：

$$\text{entropy}=-(5/14)\log_2(5/14)-(9/14)\log_2(9/14)=0.941$$

（3）计算分裂点 temperature 属性每个取值的信息熵：

分裂点为 65：

$$\text{entropy(temperature=no1)}=-(1/2)\log_2(1/2)-(1/2)\log_2(1/2)=1$$

$$\text{entropy(temperature=no2)}=-(8/12)\log_2(8/12)-(4/12)\log_2(4/12)=0.918$$

分裂点为 69：

$$\text{entropy(temperature=no1)}=-(3/4)\log_2(3/4)-(1/4)\log_2(1/4)=0.811$$

$$\text{entropy(temperature=no2)}=-(6/10)\log_2(6/10)-(4/10)\log_2(4/10)=0.971$$

分裂点为 70：