第5章　机器学习算法

本章学习目标

- 掌握机器学习模型的评价指标，能够区分有监督学习和无监督学习算法；
- 掌握机器学习中主流的分类算法；
- 掌握回归算法，并能够理解算法运算流程；
- 掌握无监督学习的聚类算法，了解主流的数据维归约技术；
- 掌握关联分析相关算法；
- 理解隐马尔可夫模型、条件随机场的理论推导，了解各种 Boosting 算法模型；
- 理解深度学习相关概念，理解深度神经网络模型。

5.1　机器学习的基本概念

5.1.1　算法分类

机器学习的核心任务是找出复杂数据隐含的规律，它的应用非常广泛，例如垃圾邮件分类、人脸识别、语音识别、医疗诊断、车牌号识别等。

在机器学习的过程中，可以按照样本数据集的特点以及学习任务的不同进行算法分类。按照数据集是否带有标签值，可以将机器学习算法分为有监督学习和无监督学习。按照数据集标签值的类型，可以将有监督学习算法进一步细分为分类问题和回归问题。在有监督学习中，根据求解方法的不同，可以将学习算法分为生成模型和判别模型。

1. 有监督学习

样本数据集带有标签值，从样本数据中学习得到一个模型，然后利用模型对新的数据进行预测推断。其数学描述为：数据集为(x,y)，其中 x 为样本的特征数据集，y 为对应的标签数据。有监督学习的目标是根据训练样本数据集确定模型 $y=f(x)$，确定函数的依据是使得函数的输出值 y 与真实标签值之间的误差最小。在有监督学习过程中，通常是在数据集中取一部分用来确定训练模型 $y=f(x)$，剩余数据用来检测模型的泛化能力。

有监督学习主要包含分类问题和回归问题。

分类问题的定义为：在有监督学习中，如果数据集的标签数据为整数，那么 $y=f(x)$就是一个由特征向量到整数的函数，这就是分类问题。通常情况下样本的标签是其类别标号，例如标号为 0 或者 1，这就是一个二分类问题。如果用 0 表示“男”，1 表示“女”，函数输出为 1，就表示样本的性别是女。分类问题中主要包含朴素贝叶斯分类、决策树、k 近邻算法、

logistic 回归算法、支持向量机算法、随机森林算法等。

回归问题的定义为：在有监督学习过程中，如果数据集的标签数据是连续的实数，那么 $y=f(x)$ 就是一个由特征向量到实数的函数，这就是回归问题。回归问题主要包含线性回归算法、决策树回归算法、随机森林回归算法等。

2. 无监督学习

在无监督学习中样本数据集不带标签值，通过对样本集的分析发现数据集的特征模式或分布规律。聚类是无监督学习的典型代表。聚类就是根据学习目的，把数据集划分为需要的几类，使得类内的样本数据相似性比较大，类间的样本数据相似性比较小。

5.1.2 模型评价指标

在学习得到机器学习算法和模型后，需要评价模型的精度，就需要定义评价模型的指标。有监督学习分为训练和预测两个阶段，通常用测试样本来检验模型的精度。在分类问题中人们用准确率来作为评价指标，它由测试样本中被正确分类的样本数与总测试样本数的比值得到。回归问题中的评价指标是回归误差，其定义方式为预测函数的输出值与样本标签值之间的均方误差。

1. 精度与召回率

分类问题中常用的两个评价指标是精度和召回率，在二分类问题中，样本分为正样本和负样本，测试样本中正样本被分类器判定为正样本的数量记为 TP，被判定为负样本的数量记为 FN，负样本中被正确分类的样本数记为 TN，被错误判定为正样本的数量记为 FP。

精度的定义为

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

召回率的定义为

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

精度是被分类器预测为正样本的数量中真正的正样本所占的比例，值越接近于 1，对正样本的分类越准确。召回率是所有正样本中被分类器判定为正样本的比例。

2. ROC 曲线

对于二分类问题，可以调整模型参数从而得到不同的分类结果，将各种参数下模型分类的准确率连成一条曲线，即 ROC 曲线，每一种模型会对应一条 ROC 曲线，可以从 ROC 曲线形状来判别模型的分类性能。首先定义真阳率和假阳率指标。真阳率是所有正样本被分类器判定为正样本的比例，即

$$\mathrm{TPR}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

假阳率是所有负样本中被分类器判定为正样本的比例，即

$$\mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$$

ROC 曲线的横轴为假阳率，纵轴为真阳率。当假阳率增加时，真阳率会增加，因此 ROC 曲线是一条增长的曲线。好的分类器要保证假阳率低，真阳率高。同一数据集下三种不同模

型对应的 ROC 曲线如图 5.1 所示。ROC 曲线下方面积越大，算法性能越好。

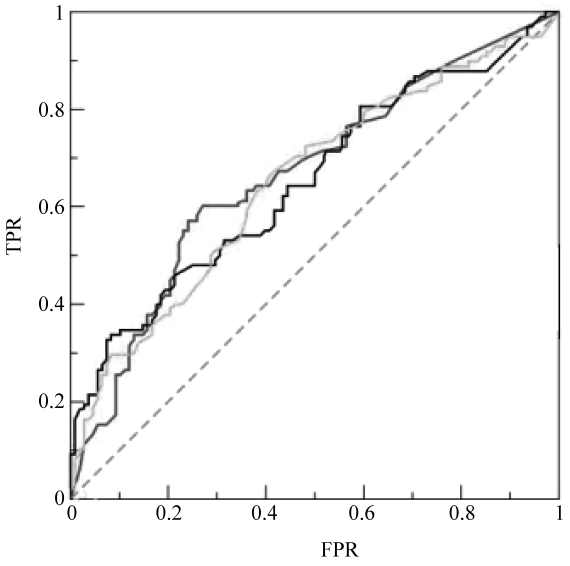

图 5.1　ROC 曲线示意图

3. 混淆矩阵

多分类问题中的准确率可以利用混淆矩阵计算。对应 k 分类问题，混淆矩阵中元素 c_{ij} 表示为第 i 类样本被分到第 j 类中的数量。

混淆矩阵可表示为

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1k} \\ c_{21} & c_{22} & \cdots & c_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{k1} & c_{k2} & \cdots & c_{kk} \end{pmatrix}$$

如果所有的样本都能被正确分类，则混淆矩阵为对角阵，所以对角线元素的和越大，分类准确率越高。

4. 交叉验证

在对精度指标计算时，常采用交叉验证法，它是把数据集中一部分作为训练集，另一部分作为测试集来统计模型的准确率。在 k 折交叉验证中，需要把数据集进行等分成 k 份，每次取 $k-1$ 份训练模型，剩余的 1 份用于检验模型的精度。这样训练模型的次数就有 C_k^{k-1} 次，用 C_k^{k-1} 个准确率的平均值来作为最终的准确率。

5.1.3　模型选择及求解问题

在给出模型的评价指标后，就导致模型误差的因素进行分析，并给出一般化的求解方案。

1. 过拟合与欠拟合问题

有监督学习的目标是模型在训练数据集上实现误差最小化，并具有较强的泛化能力。

在训练数据集上误差最小化可能会产生下面的问题：在训练数据集上模型预测误差很小，但是在测试数据集上误差很大，这就是过拟合问题；如果模型在训练数据集上表现也不好，就是欠拟合问题。欠拟合和过拟合对应模型的泛化能力都不够好。

导致欠拟合问题的原因可能有：模型过于简单；数据属性过少，不能反映数据的内在规律；非线性问题线性化等。导致过拟合问题的原因可能有：模型过于复杂；训练样本缺乏代表性；模型对噪声数据敏感等。欠拟合和过拟合的判别标准如表 5.1 所示。

表 5.1　欠拟合与过拟合的判别标准

训练集上的表现	测试集上的表现	结　　论
不好	不好	欠拟合
好	不好	过拟合
好	好	适度拟合

2. 偏差与方差分解

在衡量模型的泛化能力时常把泛化误差分解成偏差和方差。偏差是模型本身导致的误差，它是模型的预测期望值和真实值之间的差距，假设样本的特征向量为 $\boldsymbol{x}$，标签为 y，要拟合的函数为 $f(\boldsymbol{x})$，算法拟合的函数为 $\widehat{f(\boldsymbol{x})}$，则偏差定义为

$$\mathrm{Bias}(\widehat{f(\boldsymbol{x})})=E[\widehat{f(\boldsymbol{x})}-f(\boldsymbol{x})]$$

高的偏差意味着模型本身的输出和期望值差别大，模型欠拟合。方差是模型对样本数据的敏感度造成的，计算公式为

$$\mathrm{Var}(\widehat{f(\boldsymbol{x})})=E[\widehat{f(\boldsymbol{x})}^2]-E^2[\widehat{f(\boldsymbol{x})}]$$

较大的方差意味着噪声对模型影响比较大，从而出现过拟合现象。

模型的总体误差可以定义为

$$E[f(\boldsymbol{x})-\widehat{f(\boldsymbol{x})}]^2=\mathrm{Bias}(\widehat{f(\boldsymbol{x})})+\mathrm{Var}(\widehat{f(\boldsymbol{x})})+\sigma^2$$

其中 σ^2 为噪声项。

3. 正则化

所谓的正则化是对学习算法的修改，它的目的是提高模型的泛化能力，减少模型的测试误差。在机器学习中常见的正则化策略有以下 3 种：

(1) 基于先验知识在目标函数中添加约束和惩罚项；

(2) 模型选择偏好简单模型；

(3) 其他形式正则化：集成学习中采用多模型训练数据。

例如，有监督学习算法的目标是最小化误差函数，它的均方误差损失函数为

$$L(\omega)=\frac{1}{2n}\sum_{i=1}^{n}[f_\omega(x_i)-y_i]^2$$

其中，y_i 是样本 x_i 的标签值；$f_\omega(x_i)$ 为模型的输出值；ω 为模型参数。在预测函数的类型选定后，只有函数的参数 ω 可控，为了防止过拟合，可以在损失函数中加一个惩罚项，对产生的复杂模型进行惩罚，强制模型的参数尽可能简化。加入惩罚项之后的损失函数为

$$L(\omega)=\frac{1}{2n}\sum_{i=1}^{n}[f_{\omega}(x_i)-y_i]^2+\lambda r(\omega)$$

模型的后半部分 $r(\omega)$即为正则化项，λ 称为惩罚系数。

常见的正则化项有 L_1 正则和 L_2 正则，其中 L_1 正则为

$$L_1=\|\boldsymbol{\omega}\|=\sum_{i=1}^{d}|\omega_i|$$

L_2 正则为

$$L_2=\frac{1}{2}\|\boldsymbol{\omega}\|^2=\frac{1}{2}\boldsymbol{\omega}\cdot\boldsymbol{\omega}$$

对于 L_1 正则和 L_2 正则的解释：产生过拟合的原因通常是因为模型中参数比较大，添加正则项后，如果某个参数比较大，则目标函数加上正则化项后也就会变大，因此该参数就不是最优解了。

5.2　分　　类

分类任务是确定研究对象属于哪一个目标类的问题。在现实中有各种各样的分类问题，例如从邮件的标题和内容检测出垃圾邮件，从复杂的医疗影像中甄别肿瘤是恶性还是良性，从海量的客户数据库中寻找优质客户等，都是分类要解决的问题。本节介绍一些常见的分类算法。

5.2.1　k 近邻算法

1. k 近邻算法原理

k 近邻算法由 Thomas M. Cover 和 Peter E. Hart 在 1967 年提出，它的基本思想是要确定一个待测样本的类别，可以先计算它和所有训练样本之间的距离，对距离排序，找出和样本距离最小的 k 个样本，观测 k 个样本的类别，k 个样本中出现次数最多的那个类别就是对待测样本的分类结果。

例如，有两类不同的样本数据如图 5.2 所示，分别用小正方形和小三角形表示，而图正中间的圆点所表示的数据则是待分类的数据。也就是说，现在人们不知道中间圆点的数据从属于哪一类(正方形或三角形)。依据 k 近邻算法，如果选择 $k=3$，距离圆点最近的 3 个邻居是 2 个三角形和 1 个正方形，基于统计的方法，判定待分类点属于三角形一类；如果选择 $k=5$，圆点最近的 5 个邻居是 2 个三角形和 3 个正方形，基于统计的方法，判定待分类点属于正方形一类。

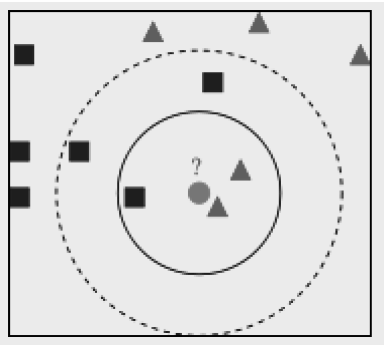

图 5.2　k 近邻算法示意图

由此可见，在 k 近邻算法中 k 值的选择非常重要，如果 k 太小，意味着只有与待测样本较近的训练样本才会对预测结果起作用，容易产生过拟合现象；如果 k 值较大，这时与待测样本较远的训练样本的标签也会对预

测起作用，使预测发生错误。在实际应用中，k 值一般选择一个较小的数值，通常采用交叉验证的方法来选择最优的 k 值。

2. 预测算法

k 近邻算法中没有具体模型，因此没有对模型的训练过程，算法唯一的参数 k 需要人工指定。它在预测时需要计算待测样本和训练样本之间的距离。

在分类问题中，假设有 n 个训练样本$(\boldsymbol{x}_i, y_i)$，其中 $\boldsymbol{x}_i$ 为样本的特征向量，y_i 为对应的标签值，给定参数 k 值，假设训练样本有 c 类，待分类的样本为 $\boldsymbol{x}$，k 近邻算法的流程如下：

(1) 计算待测样本与训练样本之间的距离，找出距离 $\boldsymbol{x}$ 最近的 k 个样本，假设由这些样本组成的集合为 D；

(2) 统计 N 中每一类样本的个数 $C_i, i=1,2,\cdots,c$；

(3) 最终的分类结果为 $\text{argmax}C_i$，其中 $\text{argmax}C_i$ 表示最大的 C_i 值对应的那个类 i。

算法 5.1　k 近邻分类算法

输入	给定 k，D 是训练样本集，待分类样本 $\boldsymbol{x}$
输出	$\boldsymbol{x}$ 类标号
方法	(1) for 每个测试样本 $z=(\boldsymbol{x}', y')$，do； (2) 计算 z 和 D 中每个样本之间的距离 d； (3) 选择离 z 最近的 k 个训练样本集合 $D_z \subset D$； (4) $y' = \underset{v}{\text{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$； (5) end for。

3. 常见距离

由于 k 近邻算法的实现需要计算样本之间的距离，不同的数据特点有不同的距离计算方式，常见的距离计算方式有下面 4 种。

(1) 欧氏距离：设 $\boldsymbol{x}, \boldsymbol{y}$ 为 n 维向量，两点之间的欧氏距离定义为

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

使用欧式距离是应将特征向量的每个分量归一化，避免特征向量分量尺度不同带来的干扰。

(2) 曼哈顿距离：设 $\boldsymbol{x}, \boldsymbol{y}$ 为 n 维向量，两点之间的曼哈顿距离定义为

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

(3) 切比雪夫距离：设 $\boldsymbol{x}, \boldsymbol{y}$ 为 n 维向量，两点之间的切比雪夫距离定义为

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

(4) 闵氏距离：设 $\boldsymbol{x}, \boldsymbol{y}$ 为 n 维向量，两点之间的闵氏距离定义为

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

4. 局限性

k 近邻算法实现简单，缺点是当训练数据样本大、特征向量的维度高时，计算的复杂度高。因为每次预测时都要计算待测样本和每一个训练样本的距离，并且需要对距离排序，找

出最近的 k 个样本。

k 近邻算法中一个需要解决的问题就是 k 值的确定，它需要根据问题和数据的特点来确定。在实现时可以考虑样本的权重，即每个样本有不同的决策权重，这对应的是带有权重的 k 近邻算法。

5.2.2　决策树算法

决策树是一种基于规则的分类方法，它用一组嵌套的规则来对待测样本进行预测，在树的每个决策结点处，根据判断结果进入下一个分支，反复操作直至到达叶子结点，得到预测结果。用于预测的规则是通过对样本训练得到的，不是人工指定的。

1. 决策树的工作原理

为了解释决策树分类原理，此处考虑脊椎动物分类问题。在这里只考虑两个类别：哺乳动物和非哺乳动物。假设生物学家发现一个新的物种，怎样判别它是哺乳动物还是非哺乳动物呢？一种方法是针对动物的特征提出一系列问题，根据问题的答案来确定。第一个问题可以为“该物种是冷血还是恒温动物？”，如果是冷血的，则该物种肯定不是哺乳动物；否则它肯定是鸟类或哺乳动物。如果是恒温动物，需要接着问，该物种是否是胎生的？如果是，则它是哺乳动物，如果不是，就是非哺乳动物。

上面的例子表明，通过提出一系列关于待测样本属性的问题，可以解决分类问题。每当一个问题得到答案，后续问题会随之而来，直到得到待测样本的类别标号。这一系列的问题和问题可能的答案可以组成决策树的形式。决策树是一种由结点和有向边组成的层次结构。哺乳动物分类问题的决策树如图 5.3 所示。

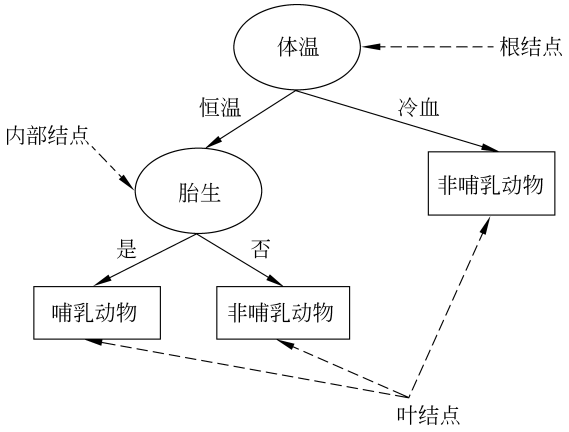

图 5.3　哺乳动物分类的决策树

决策树中包含 3 种结点：

(1) 根结点，没有入边，但有零条或多条出边。

(2) 内部结点，恰好有一条入边和两条或多条出边。

(3) 叶结点，恰好有一条入边，没有出边。

在决策树中根结点和内部结点都是决策结点，在这些结点处需要进行判别以决定进入

哪个分支。决策树中的每个叶结点都赋予一个类标号，在决策结点处会设置属性判别条件，以用来区别具有不同属性特性的样本。在上述例子中使用体温这个属性把冷血脊椎动物和恒温脊椎动物区别开，因为所有的冷血动物都是非哺乳动物，所以有一个类标号非哺乳动物的叶结点来作为根结点的右子结点，如果脊椎动物是恒温的，则接下来用是否胎生这个内部结点来区分哺乳动物和其他恒温动物。

一旦决策树构造完成，对待测样本进行分类变得相当简单。从根结点开始，将待测样本根据判别条件选择适当的分支，沿着分支到达一个内部结点或者叶结点，在内部结点使用新的判别条件，进入下一个结点，直至到达叶结点，叶结点的类标号就是待测样本的类标号。例如，用上例中的决策树预测火烈鸟的类别标号所经过的路径，路径终止于类标号为非哺乳动物的叶结点。对未知类别的火烈鸟利用决策树分类的过程如图 5.4 所示。

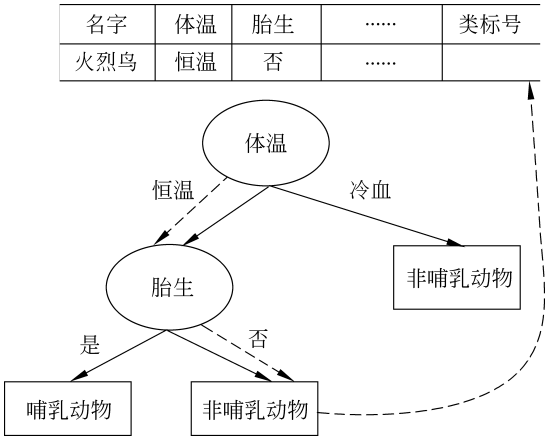

图 5.4　对未知类别的火烈鸟利用决策树分类的过程

2. 如何建立决策树

理论上来讲，对于给定的属性集，可以构造决策树的数目达指数级。尽管有些决策树比其他决策树更准确，但是由于搜索空间是指数规模的，在计算上是不可行的。常见的决策树构造过程都是基于局部最优的策略来选择划分数据的属性的，Hunt 算法就是这样一种算法。它是 ID3、C4.5 和 CART 等决策树算法的基础。

在 Hunt 算法中，通过将训练样本相继划分子集，以递归方式建立决策树。设 D_t 是与结点 t 相关联的训练样本集，$\{y_1, y_2, \cdots, y_c\}$ 是类标号，Hunt 算法的流程如下。

(1) 如果 D_t 中所有记录都属于同一个类 y_t，则 t 是叶结点，用 y_t 标记。

(2) 如果 D_t 中包含属于多个类的记录，则选择一个属性判别条件，将样本集划分成较小的子集。对于判别条件的每一个答案，都创建一个子结点，并根据判别结果将 D_t 中的样本分配到子结点中。然后对于每个子结点递归调用该算法。

决策树算法必须要解决以下两个问题。

(1) 如何分裂训练样本。在树的增长过程中每一次递归都必须选择一个属性作为判别条件，将训练样本集划分成较小的子集。为了实现这个步骤，算法必须提供不同类型的属性判别条件方法，并且提供判别属性选择的度量方法。

(2) 如何停止分裂过程。决策树的生成过程中需要有终止树生成的条件，一种终止条件是结点处的所有训练样本具有相同的属性；另一种终止条件是结点处的样本数量小于一个阈值时，终止树的生成。

3. 属性的种类及对应输出

决策树算法中不同类型的样本属性提供不同的判别条件和对应的输出。样本属性可以有如下 4 种。

(1) 二元属性：属性取值只有两个，属性判别产生两个可能输出，例如性别属性会有“男”“女”两个可能输出。

(2) 多元属性：样本属性有多个可能取值，属性判别后可以有多个输出。例如婚姻状况可能有 3 个属性值“未婚”“已婚”“离异”，由此会产生一个三路划分。

(3) 序数属性：序数属性可以产生二元或多元划分，例如鞋子的号码、衣服的尺码等，都是属于序数属性。

(4) 连续属性：属性的取值是连续值，对于连续属性可以根据判别条件设定二元输出($A<v$，或者 $A\geqslant v$)，也可以依据属性的取值区间进行离散化，设置多元输出。

4. 最佳分裂的度量

在决策结点处需要找到一个分裂规则来将训练样本划分成两个子集(即在训练样本集中选择一个属性作为分裂规则)，因此要确定属性选择的标准，根据标准寻找最佳的分裂。对于分类问题，要保证每次分裂之后的左右子结点里面的样本尽可能纯，即子结点之间的样本类别不相交。为此，需要定义不纯度指标，当样本都属于某一类时，其不纯度为 0，当样本均匀地属于所有类时，不纯度最大。满足这个条件的不纯度度量指标主要有信息熵不纯度、Gini 系数不纯度、分类不纯度等。

不纯度指标都是由样本集每类样本出现的概率来构造的，要先计算样本集中每个类别出现的概率 $p_i=\dfrac{N_i}{N}$，其中 N_i 是第 i 类样本数，N 为总样本数。

测试样本集 D 的信息熵不纯度定义为 $E(D)=-\sum\limits_i p_i\log_2 p_i$，当样本只属于一类时熵最小，当样本均匀地分布在所有类中时熵最大。

Gini 系数不纯度的定义为 $G(D)=1-\sum\limits_i p_i^2$，当样本属于同一类时，Gini 系数不纯度最小，最小值为 0；当样本均匀地分布在所有类中时，Gini 系数不纯度最大。

分类不纯度的定义为 $E(D)=1-\max(p_i)$，当样本属于同一类时，分类不纯度最小，最小值为 0；当样本均匀地分布在所有类中时，分类不纯度最大。

最佳的分裂是指在结点处将训练样本分成左右两个子集，两个子集要尽量纯，也就是要求分裂之后的两个子集的不纯度指标之和最小。根据样本的不纯度，还可以定义分裂的不纯度，其一般形式为

$$G=\frac{N_{\mathrm{L}}}{N}G(D_{\mathrm{L}})+\frac{N_{\mathrm{R}}}{N}G(D_{\mathrm{R}})$$

其中，$G(D_{\mathrm{L}})$是左子集的不纯度；$G(D_{\mathrm{R}})$是右子集的不纯度；N_{L} 是左子集的样本数；N_{R} 是右子集的样本数；N 是总样本数。

如果采用信息熵不纯度，对应的计算公式为

$$G=\frac{N_{\mathrm{L}}}{N}\left(-\sum_{i}\frac{N_{\mathrm{L},i}}{N_{\mathrm{L}}}\log_2\frac{N_{\mathrm{L},i}}{N_{\mathrm{L}}}\right)+\frac{N_{\mathrm{R}}}{N}\left(-\sum_{i}\frac{N_{\mathrm{R},i}}{N_{\mathrm{R}}}\log_2\frac{N_{\mathrm{R},i}}{N_{\mathrm{R}}}\right)$$

其中，$N_{\mathrm{L},i}$是左子集中第i类的样本数；$N_{\mathrm{R},i}$是右子集中第i类的样本数。

如果采用 Gini 系数不纯度，对应的计算公式变为

$$\begin{aligned}G&=\frac{N_{\mathrm{L}}}{N}\left(1-\frac{\sum_i N_{\mathrm{L},i}^2}{N_{\mathrm{L}}}\right)+\frac{N_{\mathrm{R}}}{N}\left(1-\frac{\sum_i N_{\mathrm{R},i}^2}{N_{\mathrm{R}}}\right)\\&=\frac{1}{N}\left(N_{\mathrm{L}}-\frac{\sum_i N_{\mathrm{L},i}^2}{N_{\mathrm{L}}}+N_{\mathrm{R}}-\frac{\sum_i N_{\mathrm{R},i}^2}{N_{\mathrm{R}}}\right)=1-\frac{1}{N}\left(\frac{\sum_i N_{\mathrm{L},i}^2}{N_{\mathrm{L}}}+\frac{\sum_i N_{\mathrm{R},i}^2}{N_{\mathrm{R}}}\right)\end{aligned}$$

其中，$N_{\mathrm{L},i}$是左子集中第i类的样本数；$N_{\mathrm{R},i}$是右子集中第i类的样本数。

生成决策树时，先选择不纯度度量指标，按照计算公式计算不纯度值，能够使得不纯度最小的的分裂就是最佳分裂。

ID3 算法的核心思想是在决策树的每一个非叶子结点划分之前，先计算每一个特征所带来的信息增益，选择其中最大信息增益的特征来划分该结点，因为信息增益越大，区分样本的能力就越强，越具有代表性，信息增益的计算公式为

$$\begin{aligned}\mathrm{Gain}(D,A)&=E(D)-\left[\frac{N_{\mathrm{L}}}{N}G(D_{\mathrm{L}})+\frac{N_{\mathrm{R}}}{N}G(D_{\mathrm{R}})\right]\\&=-\sum_i p_i\log_2 p_i+\frac{N_{\mathrm{L}}}{N}\left(\sum_i\frac{N_{\mathrm{L},i}}{N_{\mathrm{L}}}\log_2\frac{N_{\mathrm{L},i}}{N_{\mathrm{L}}}\right)+\frac{N_{\mathrm{R}}}{N}\left(\sum_i\frac{N_{\mathrm{R},i}}{N_{\mathrm{R}}}\log_2\frac{N_{\mathrm{R},i}}{N_{\mathrm{R}}}\right)\end{aligned}$$

其中，A表示训练样本属性，若B，C为样本的其他属性，比较 $\mathrm{Gain}(D,A)$，$\mathrm{Gain}(D,B)$，$\mathrm{Gain}(D,C)$的大小，按照属性信息增益大的属性进行分裂。

在 CART 决策树算法中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。在 CART 算法中，用 Gini 系数不纯度来选择最优分裂属性，计算公式为

$$\mathrm{Gini}(D,A)=\frac{N_{\mathrm{L}}}{N}\left(1-\frac{\sum_i N_{\mathrm{L},i}^2}{N_{\mathrm{L}}}\right)+\frac{N_{\mathrm{R}}}{N}\left(1-\frac{\sum_i N_{\mathrm{R},i}^2}{N_{\mathrm{R}}}\right)$$

C4.5 算法是对 ID3 算法的改进，ID3 算法是用信息增益来选择特征的，而信息增益的缺点是比较偏向选择取值较多的特征，如果有些特征取值比其他特征的取值多很多，这个特征基本就直接被认为是最重要的特征，但实际却未必。为改善这种情况，在 C4.5 中，引入对取值数量的惩罚，得到信息增益率作为特征重要性的衡量标准。如果某一个特征取值越多，那么对它的惩罚越严重，其信息增益率也能得到控制。信息增益率计算公式为

$$\begin{aligned}\mathrm{Gain}(D,A)\_\mathrm{Ratio}&=\frac{\mathrm{Gain}(D,A)}{G(D,A)}\\&=\frac{\mathrm{Gain}(D,A)}{\frac{N_{\mathrm{L}}}{N}\left(-\sum_i\frac{N_{\mathrm{L},i}}{N_{\mathrm{L}}}\log_2\frac{N_{\mathrm{L},i}}{N_{\mathrm{L}}}\right)+\frac{N_{\mathrm{R}}}{N}\left(-\sum_i\frac{N_{\mathrm{R},i}}{N_{\mathrm{R}}}\log_2\frac{N_{\mathrm{R},i}}{N_{\mathrm{R}}}\right)}\end{aligned}$$