决策树及其几种常用算法

王裕 经济系

2019/05/05

摘要

在课堂上我们已经学习了决策树的相关原理以及 CART 算法，本文将回顾决策树的原理，并进一步介绍决策树的几种常用算法。

一、决策树的原理

1、决策树的定义

决策树（Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。在机器学习中，决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。

分类树（决策树）是一种十分常用的分类方法。他是一种监管学习，所谓监管学习就是给定一堆样本，每个样本都有一组属性和一个类别，这些类别是事先确定的，那么通过学习得到一个分类器，这个分类器能够对新出现的对象给出正确的分类。这样的机器学习就被称之为监督学习。

下面主要讨论用与分类的决策树。决策树呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程。学习时，利用训练数据，根据损失函数最小化的原则建立决策树模型；预测时，对新的数据，利用决策模型进行分类。

决策树可以分为两类，主要取决于它目标变量的类型。

离散性决策树：离散性决策树，其目标变量是离散的，如性别：男或女等；

连续性决策树：连续性决策树，其目标变量是连续的，如工资、价格、年龄等；

2、决策树的重要概念

（1）根结点(Root Node)：它表示整个样本集合，并且该节点可以进一步划分成两个或多个子集。

（2）拆分(Splitting)：表示将一个结点拆分成多个子集的过程。

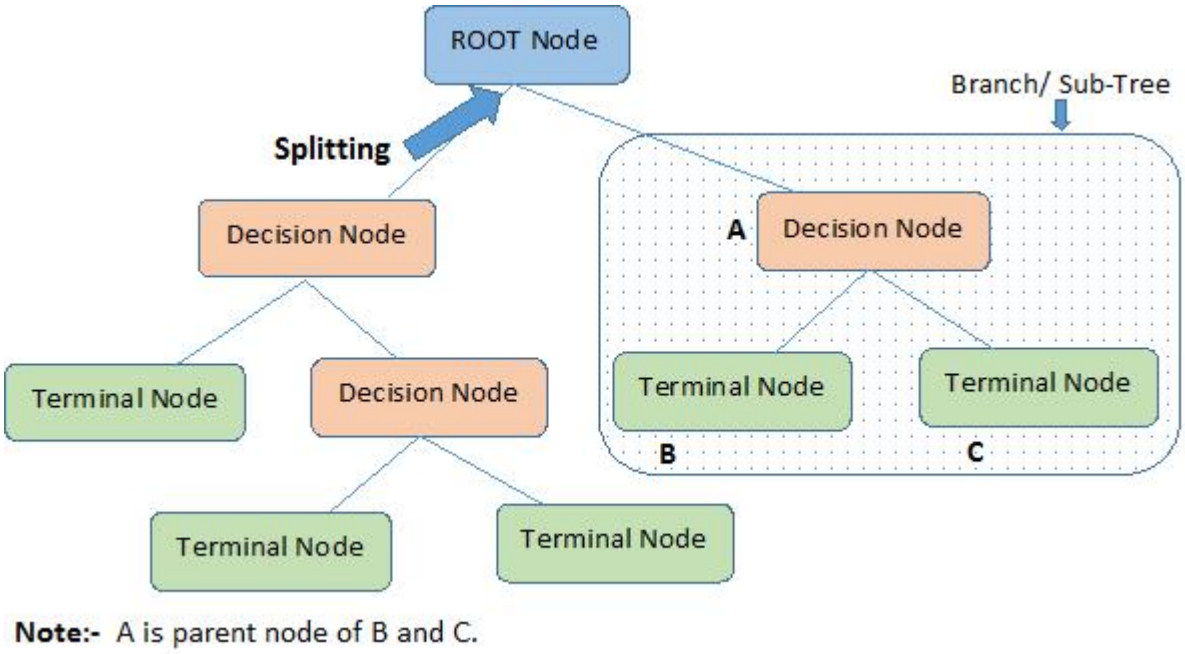
（3）决策结点(Decision Node)：当一个子结点进一步被拆分成多个子节点时，这个子节点就叫做决策结点。

（4）叶子结点(Leaf/Terminal Node)：无法再拆分的结点被称为叶子结点。

（5）剪枝(Pruning)：移除决策树中子结点的过程就叫做剪枝，跟拆分过程相反。

（6）分支/子树(Branch/Sub-Tree)：一棵决策树的一部分就叫做分支或子树。

（7）父结点和子结点(Paren and Child Node)：一个结点被拆分成多个子节点，这个结点就叫做父节点；其拆分后的子结点也叫做子结点。



3、决策树的构造过程

决策树的构造过程一般分为 3 个部分，分别是特征选择、决策树生成和决策树裁剪。

（1）特征选择

特征选择表示从众多的特征中选择一个特征作为当前节点分裂的标准，如何选择特征有不同的量化评估方法，从而衍生出不同的决策树，如 ID3（通过信息

增益选择特征）、C4.5（通过信息增益比选择特征）、CART（通过 Gini 指数选择特征）等。

目的（准则）：使用某特征对数据集划分之后，各数据子集的纯度要比划分钱的数据集 D 的纯度高（也就是不确定性要比划分前数据集 D 的不确定性低）

（2）决策树的生成

根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。这个过程实际上就是使用满足划分准则的特征不断的将数据集划分成纯度更高，不确定行更小的子集的过程。对于当前数据集的每一次划分，都希望根据某个特征划分之后的各个子集的纯度更高，不确定性更小。

（3）决策树的裁剪

决策树算法很容易过拟合（overfitting），剪枝算法就是用来防止决策树过拟合，提高泛化性能的方法。

剪枝分为预剪枝与后剪枝。

预剪枝是指在决策树的生成过程中，对每个节点在划分前先进行评估，若当前的划分不能带来泛化性能的提升，则停止划分，并将当前节点标记为叶节点。

后剪枝是指先从训练集生成一颗完整的决策树，然后自底向上对非叶节点进行考察，若将该节点对应的子树替换为叶节点，能带来泛化性能的提升，则将该子树替换为叶节点。

那么怎么来判断是否带来泛化性能的提升那？最简单的就是留出法，即预留一部分数据作为验证集来进行性能评估。

4、决策树的优缺点

4.1 优点

易于理解和解释，决策树可以可视化。

几乎不需要数据预处理。其他方法经常需要数据标准化，创建虚拟变量和删除缺失值。决策树还不支持缺失值。

使用树的花费（例如预测数据）是训练数据点(data points)数量的对数。

可以同时处理数值变量和分类变量。其他方法大都适用于分析一种变量的集合。

可以处理多值输出变量问题。

使用白盒模型。如果一个情况被观察到，使用逻辑判断容易表示这种规则。相反，如果是黑盒模型（例如人工神经网络），结果会非常难解释。

即使对真实模型来说，假设无效的情况下，也可以较好的适用。

4.2 缺点

决策树学习可能创建一个过于复杂的树，并不能很好的预测数据。也就是过拟合。修剪机制（现在不支持），设置一个叶子节点需要的最小样本数量，或者数的最大深度，可以避免过拟合。

决策树可能是不稳定的，因为即使非常小的变异，可能会产生一颗完全不同的树。这个问题通过 decision trees with an ensemble 来缓解。

学习一颗最优的决策树是一个 NP-完全问题 under several aspects of optimality and even for simple concepts。因此，传统决策树算法基于启发式算法，例如贪婪算法，即每个节点创建最优决策。这些算法不能产生一个全家最优的决策树。对样本和特征随机抽样可以降低整体效果偏差。

概念难以学习，因为决策树没有很好的解释他们，例如，XOR, parity or multiplexer problems。

如果某些分类占优势，决策树将会创建一棵有偏差的树。因此，建议在训练之前，先抽样使样本均衡。

二、常见决策树算法（ID3、C4.5、CART）

最早提及决策树思想的是 Quinlan 在 1986 年提出的 ID3 算法和 1993 年提出的 C4.5 算法，以及 Breiman 等人在 1984 年提出的 CART 算法。下面分别介绍这三种算法的原理。

1、ID3 算法原理

1.1 信息增益

（1）熵

在信息论中，熵（entropy）是随机变量不确定性的度量，也就是熵越大，则随机变量的不确定性越大。设 X 是一个取有限个值得离散随机变量，其概率分布为：

$$P(X = x_i) = p_i, i = 1,2,3...,n$$

则随机变量 X 的熵定义为：

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

（2）条件熵

设有随机变量（X, Y），其联合概率分布为：

$$P(X = \mathrm{x}_i, Y = \mathrm{y}_i) = p_{ij}, i = 1,2,...n, j = 1,2,...,m$$

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 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：

$$H(Y \mid X) = \sum_{i=1}^{n} p_i H(Y \mid X = x_i) \text{ 其中 } p_i = P(X = x_i), i = 1,2,3...,n$$

当熵和条件熵中的概率由数据估计得到时（如极大似然估计），所对应的熵与条件熵分别称为经验熵和经验条件熵。

（3）信息增益

定义：信息增益表示由于得知特征 A 的信息后儿时的数据集 D 的分类不确定性减少的程度，定义为：

Gain(D,A) = H(D) - H(D|A)

即集合 D 的经验熵 H(D)与特征 A 给定条件下 D 的经验条件熵 H(H|A)之差。

理解：选择划分后信息增益大的作为划分特征，说明使用该特征后划分得到的子集纯度越高，即不确定性越小。因此我们总是选择当前使得信息增益最大的特征来划分数据集。

缺点：信息增益偏向取值较多的特征（原因：当特征的取值较多时，根据此特征划分更容易得到纯度更高的子集，因此划分后的熵更低，即不确定性更低，因此信息增益更大）

1.2 ID3 算法

输入：训练数据集 D，特征集 A，阈值 ε；

输出：决策树 T。

Step1：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作为该节点的类标记，返回 T；

Step2：若 A=Ø，则 T 为单结点树，并将 D 中实例数最大的类C_K作为该节点的类标记，返回 T；

Step3：否则，2.1.1（3）计算 A 中个特征对 D 的信息增益，选择信息增益最大的特征A_K；

Step4：如果的信息增益A_g小于阈值 ε，则 T 为单节点树，并将 D 中实例数最大的类C_K作为该节点的类标记，返回 T；

Step5：否则，对A_g的每一种可能值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，由结点及其子树构成树 T，返回 T；

Step6：对第 i 个子节点，以D_i为训练集，以A-$\{A_g\}$为特征集合，递归调用 Step1-Step5，得到子树T_i，返回T_i；

2、C4.5 算法原理

C4.5 算法与 ID3 算法很相似，C4.5 算法是对 ID3 算法做了改进，在生成决策树过程中采用信息增益比来选择特征。

2.1 信息增益比

我们知道信息增益会偏向取值较多的特征，使用信息增益比可以对这一问题进行校正。

定义：特征 A 对训练数据集 D 的信息增益比 GainRatio(D,A)定义为其信息增益 Gain(D,A)与训练数据集 D 的经验熵 H(D)之比：

$$GainRatio(D,A) = \frac{Gain(D,A)}{H(D)}$$

2.2 C4.5 算法

C4.5 算法过程跟 ID3 算法一样，只是选择特征的方法由信息增益改成信息增益比。

3、CART 算法原理

3.1 Gini 指数

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：

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

备注：p_k 表示选中的样本属于 k 类别的概率，则这个样本被分错的概率为（1-p_k）。

对于给定的样本集合 D，其基尼指数为：

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

备注：这里 C_k 是 D 中属于第 k 类的样本自己，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 D1 和 D2 两部分，即：

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

则在特征 A 的条件下，集合 D 的基尼指数定义为：

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

基尼指数 Gini(D)表示集合 D 的不确定性，基尼指数 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，这一点跟熵相似。

3.2 CART 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，构建二叉树：

Step1：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 A=a 的测试为“是”或“否”将 D 分割为 D1 和 D2 两部分，利用上式 Gini(D,A)来计算 A=a 时的基尼指数。

Step2：在所有可能的特征 A 以及他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可能的切分点作为最有特征与最优切分点。依最优特征与最有切分点，从现结点生成两个子节点，将训练数据集依特征分配到两个子节点中去。

Step3：对两个子结点递归地调用 Step1、Step2，直至满足条件。

Step4：生成 CART 决策树

算法停止计算的条件是节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，或者没有更多特征。

参考文献：

https://blog.csdn.net/jiaoyangwm/article/details/79525237#36__1626

https://baike.baidu.com/item/%E5%86%B3%E7%AD%96%E6%A0%91/10377049?fr=aladdin

https://shuwoom.com/?p=1452