决策树学习笔记（三）：CART算法，决策树总结

原创 wLsq Python数据科学 2019-01-14

收录于话题
#Python数据科学 60 #机器学习 18

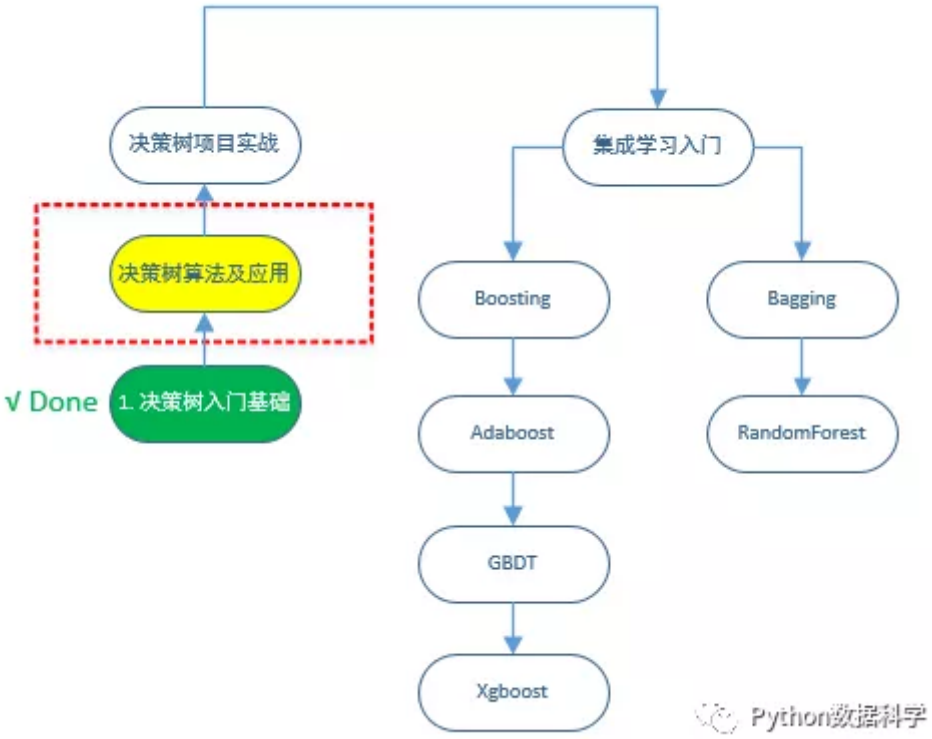
点击上方**“Python数据科学”**，选择**“星标公众号”**

关键时刻，第一时间送达！

作者：xiaoyu

介绍：一个半路转行的数据挖掘工程师

推荐导读：本篇为树模型系列第三篇，旨在从最简单的决策树开始学习，循序渐进，最后理解并掌握复杂模型GBDT，Xgboost，为要想要深入了解机器学习算法和参加数据挖掘竞赛的朋友提供帮助。



前情回顾

前两篇介绍了决策树主要的三个步骤，以及ID3和C4.5算法：

- 决策树学习笔记（一）：特征选择
- 决策树学习笔记（二）：剪枝，ID3，C4.5

本篇将继续介绍决策的第三种算法：CART算法，它可以说是学习决策树的核心了。高级集成学习很多复杂框架都是基于CART的。下面将详细介绍CART算法的来龙去脉。

- CART生成算法
- CART剪枝算法
- CART算法小结
- 决策树算法优缺点总结

CART生成算法

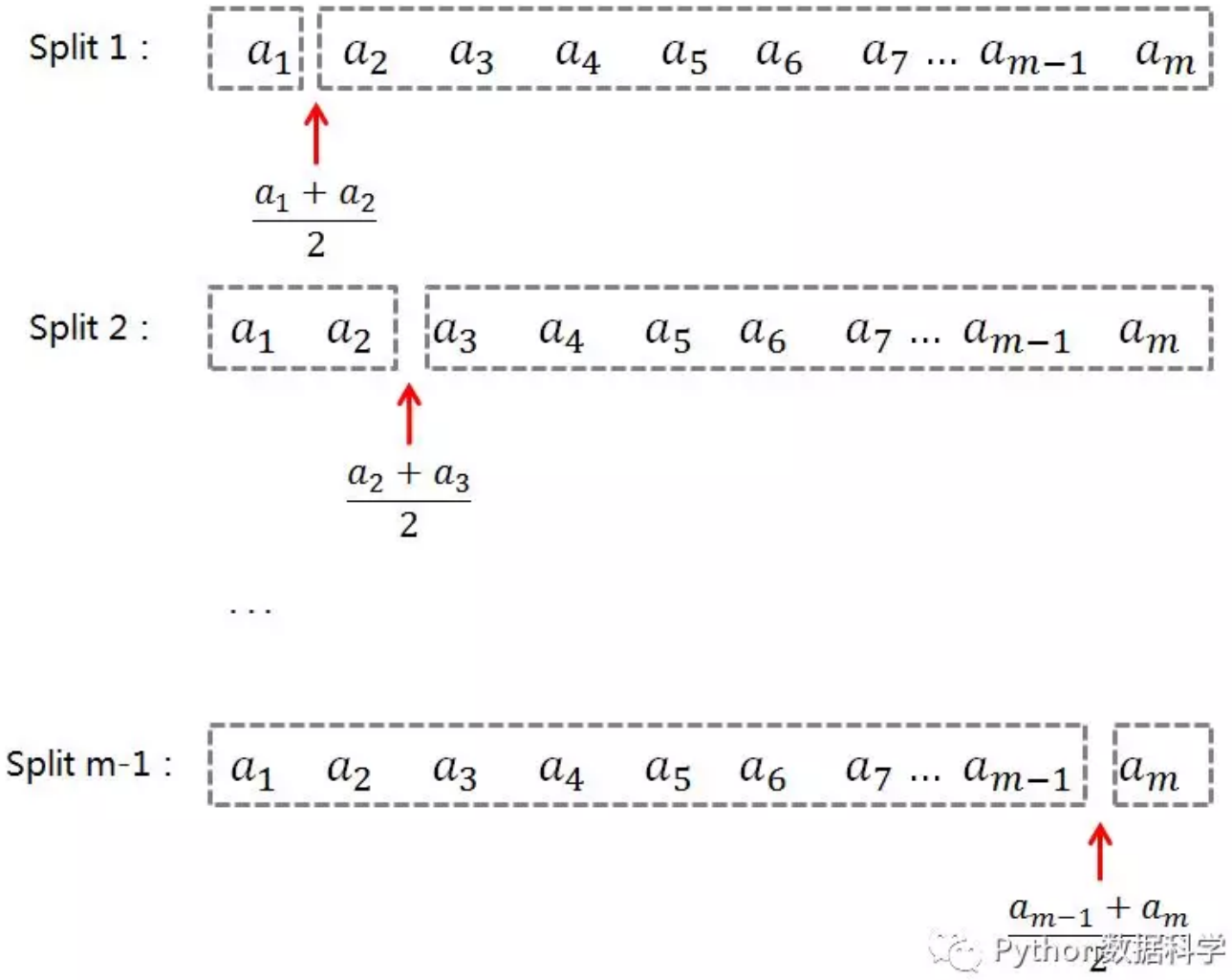
为什么叫CART算法呢？这还要从它的英文单词说起。CART是 "Classification and Regression Trees" 的缩写，意思是 **"分类回归树"**。从它的名字上就不难理解了，CART算法是既可以用于分类的，也可以用于回归的。

很多朋友诧异于决策树为什么可以用于回归，明明是if-then结构用于分类的。下面我们来分别介绍CART分类和回归两种情况。

分类树生成算法

CART算法的分类树是与ID3和C4.5有所不同。下面我们针对特征值的类型来分别介绍CART算法是如何进行分类的，以及和C4.5有什么异同。

如果特征值是连续值：CART的处理思想与C4.5是相同的，即将连续特征值**离散化。**唯一不同的地方是度量的标准不一样，CART采用基尼指数，而C4.5采用信息增益比。下面举个例子说明下：



特征a有连续值m个，从小到大排列。m个数值就有m-1个切分点，分别使用每个切分点把连续数值离散划分成两类，将节点数据集按照划分点分为D1和D2子集，然后计算每个划分点下对应的基尼指数，对比所有基尼指数，选择值最小的一个作为最终的特征划分。

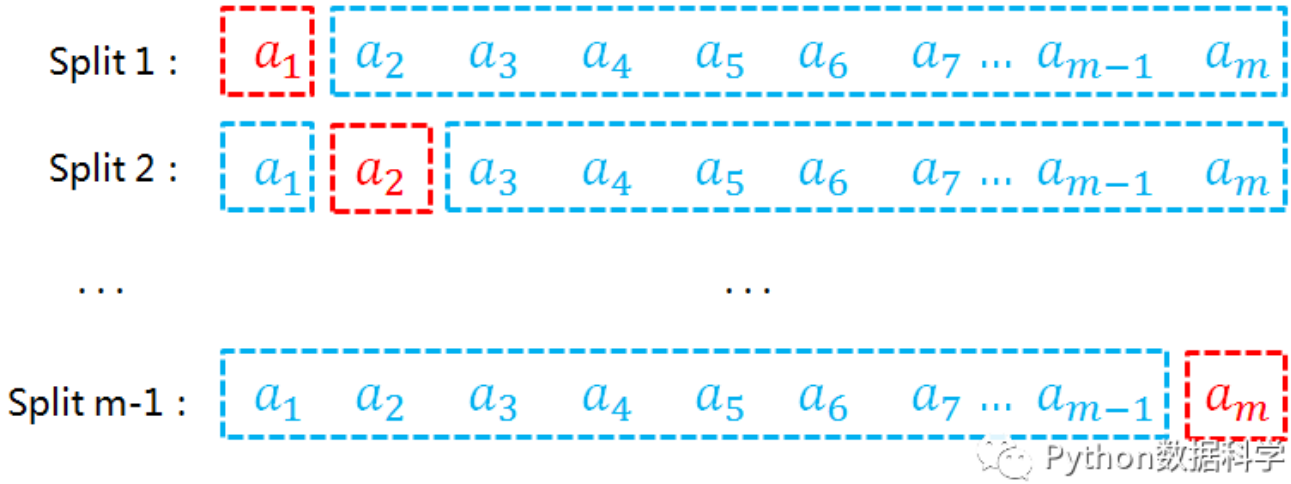
$$Gini(D) = 1 - \sum_{k=1}^{K}(\frac{|C_k|}{|D|})^2$$

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

基尼指数公式，以及基于特征A划分后的基尼指数

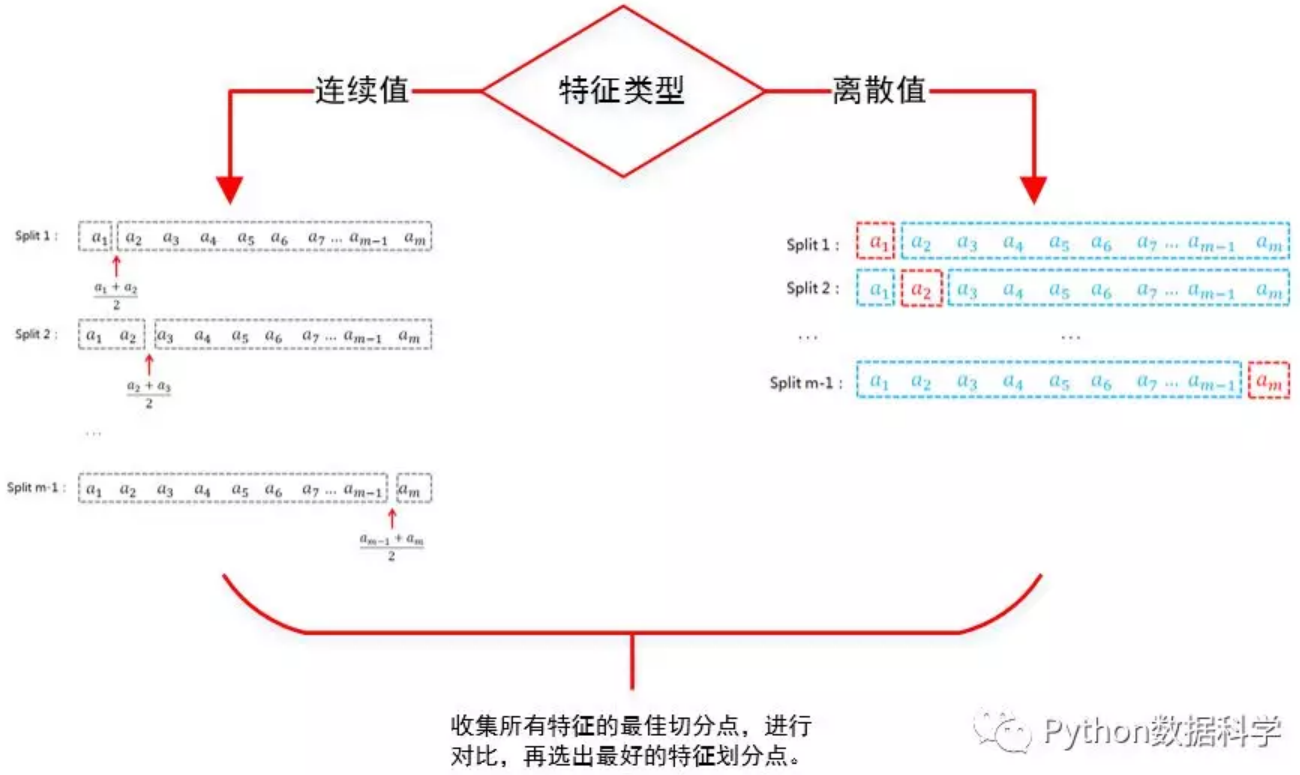
以上就实现了将连续特征值离散化，但是CART与ID3，C4.5处理离散属性不同的是：如果当前节点为连续属性，则该属性（剩余的属性值）后面还可以参与子节点的产生选择过程。

如果特征值是离散值：CART的处理思想与C4.5稍微所有不同。如果离散特征值多于两个，那么C4.5会在节点上根据特征值划分出多叉树。但是CART则不同，无论离散特征值有几个，在节点上都划分成二叉树。CART树是如何进行分类的呢？



还是假设特征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，其它非该特征分为另外一类。依照这个标准遍历所有的分类情况，计算每种分类下的基尼指数，最后选择值最小的一个作为最终的特征划分。

特征值连续和离散有各自的处理方法，不应该混淆使用。比如分类0,1,2只代表标签含义，如果进行加减的运算或者求平均则没有任何意义。因此，CART分类树会根据特征类型选择不同的划分方法，并且与C4.5不同是，它永远只有两个分支。



李航“统计学习方法”中的分类树算法流程仅仅是针对特征是离散型的情况，并没有提及连续值的情况。本篇根据上面我们介绍两个特征类型情况重新给出一个算法流程（主要就是区分两种不同特征类型）：

输入：训练数据集D，停止计算的参数条件。
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：如果样本个数小于阈值或者没有特征，
　　则返回决策子树，当前节点停止递归。
2：计算样本集D的基尼系数，如果基尼系数小于阈值，
　　则返回决策树子树，当前节点停止递归。
3：识别各个特征类型，离散值还是连续值？
　　对每种类型使用相应的处理方法并计算每个切分下的基尼系数。
　　缺失值的处理方法和C4.5算法里描述的相同。
4：在计算出来的各个特征的各个特征值对数据集D的基尼系数中，
　　选择基尼系数最小的特征A和对应的特征值a。
　　根据这个最优特征和最优特征值，把数据集划分成两部分D1和D2，
　　同时建立当前节点的左右节点，做节点的数据集D为D1，右节点的数据集D为D2.
5：对左右的子节点递归的调用1-4步，生成决策树。

算法停止计算的条件是：如步骤1,2中所示，结点中的样本个数小于预定阈值，
或样本集的Gini系数小于预定阈值（样本基本属于同一类），或者没有更多特征。

回归树生成算法

与分类树不同，回归树的预测变量是连续值，比如预测一个人的年龄，又或者预测季度的销售额等等。另外，回归树在**选择特征的度量标准**和**决策树建立后预测的方式**上也存在不同。

预测方式

一个回归树对应着输入特征空间的一个划分，以及在划分单元上的输出值。先假设数据集已被划分，R1,R2,...,Rm共m的子集，回归树要求每个划分Rm中都对应一个固定的输出值cm。

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

这个cm值其实就是每个子集中所有样本的目标变量y的平均值，并以此cm作为该子集的预测值。所有分支节点都是如此，叶子节点也不例外。因此，可以知道回归树的预测方式是：**将叶子节点中样本的y均值作为回归的预测值。**而分类树的预测方式则是：叶子节点中概率最大的类别作为当前节点的预测类别。

选择特征的度量标准

CART回归树对于特征类型的处理与分类树一样，连续值与离散值分开对待，并只能生成二叉树。但是CART回归树对于选择特征的度量标准则完全不同。

分类树的特征选择标准使用基尼指数，而回归树则使用RSS**残差平方和**。了解线性回归的朋友知道，损失函数是以最小化离差平方和的形式给出的。回归树使用的度量标准也是一样的，通过最小化残差平方和作为判断标准，公式如下：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_2\in R_2(j,s)}(y_i-c_2)^2\right]$$

> 注意：计算的是属性划分下样本的目标变量y的残差平方和，而非属性值。

- yi：样本目标变量的真实值。
- R1&R2：被划分的两个子集，回归树是二叉树，固只有两个子集。
- c1&c2：R1&R2子集的样本均值。
- j：当前的样本特征
- s：划分点

上面公式的含义是：计算所有的特征以及相应所有切分点下的残差平方和，找到一组(特征j，切分点s)，以满足：分别最小化左子树和右子树的残差平方和，并在此基础上再次最小化二者之和。

其实，回归树也有分类的思想。所谓**“物以类聚”**，相同类之间的目标变量值才会更接近，方差值也就会更小。对于回归树的生成算法，除了以上两点外，其它都分类树是相同的。

CART剪枝算法

对于决策树剪枝，上一篇已经介绍：决策树学习笔记（二）：剪枝，ID3，C4.5。这部分重点介绍下CART是如何剪枝的？选择的是哪种方式？

CART回归树和CART分类树的剪枝策略除了在度量损失的时候一个使用均方差，一个使用基尼系数，算法基本完全一样，因此将它们统一来说。CART采用的办法是**后剪枝法**，即先生成决策树，然后产生所有可能的剪枝后的CART树，然后使用**交叉验证**来检验各种剪枝的效果，选择泛化能力最好的剪枝策略。

也就是说，CART树的剪枝算法可以概括为两步：
1）是从原始决策树生成各种剪枝效果的决策树序列。
2）是用交叉验证来检验剪枝后的预测能力，选择泛化预测能力最好的剪枝后的数作为最终的CART树。

1）生成决策树序列

CART采用CCP（代价复杂度）的后剪枝方法，定义了决策树的损失函数和正则化项。公式如下：

$$C_\alpha(T) = C(T) + \alpha|T|$$

- T：决策树中的任意一个节点
- |T|：叶子节点数
- alpha：正则化参数，惩罚系数
- C(T)：无惩罚项情况下的预测误差，比如基尼指数
- C_alpha(T)：在正则参数alpha情况下节点T对应的预测误差

CART剪枝与C4.5有所不同，C4.5剪枝算法是人为给定一个alpha，然后从叶结点逐渐向根节点回溯，然而CART**多了一个遍历alpha的步骤，从0~+无穷。**

我们先明确几个概念，然后将这几个概念结合起来就可以理解整个生成决策树序列的算法流程了。

现假设我选定了决策树的任意一个节点t，并将节点t作为根节点。那么这个时候我想知道节点t下的分支是否需要剪枝，怎么办呢？

很容易想到，我们可以**对比一下剪枝以后的预测误差和剪枝以前的预测误差值的大小**，如果不剪枝的误差比剪枝的大，那么我们就执行剪枝。用公式来抽象描述一下：

$$C_\alpha(t) = C(t) + \alpha|t|$$
$$C_\alpha(t) = C(t) + \alpha$$

以t为单节点树的损失函数，|t|=1（剪枝后）

$$C_\alpha(T_t) = C(T_t) + \alpha|T_t|$$

以t为根节点的Tt损失函数（剪枝前）

现在我们有了以t为根节点剪枝前后的损失函数，我们只需要对比一下就知道了。由于alpha未确定，因此临界的情况是：

$$C_\alpha(T_t) = C_\alpha(t)$$
$$C(T_t) + \alpha|T_t| = C_\alpha(t) = C(t) + \alpha$$
$$\alpha = \frac{C_\alpha(t) - C_\alpha(T_t)}{|T_t| - 1}$$

我们把这时候的alpha临界值称为**误差增益率**，用g(t)来表示，公示如下：

$$g(t) = \frac{C_\alpha(t) - C_\alpha(T_t)}{|T_t| - 1}$$

我们可以将g(t)简单的理解为一种**阈值**，如果alpha**大于或者等于**g(t)，那么就剪枝。因为在相等的情况下，不剪枝和剪枝达到同样的效果，也就相当于这些分支没有什么作用。如果alpha小于g(t)，则保留，不剪枝。

考虑另外一个问题，**如何选择用哪个节点进行剪枝呢？**

我们上面已经找到了对某个节点下是否该剪枝的方法了，但我们开始假设的是任意一个节点t，是一个通用的方法。对于一个生成完整的决策树而言，是至少拥有一个节点的。如果一个决策树有n个节点，那么我就会有相应的**n个误差增益率g(t)**。

现在alpha是未知的，我们需要从零开始遍历，直到正无穷。显然，如果节点下的g(t)越小，alpha就会越先达到该节点的阈值，而此时的alpha大小还不足以达到其它节点的阈值。**这说明g(t)越小的节点应该越先被剪枝。**

如果我们将所有g(t)排序，g1(t)，g2(t)，…，gn(t)，那么我就会先对g1(t)对应的节点剪枝，得到一个最优子树，和alpha区间。然后在此基础上再对g2(t)对应的节点进行剪枝，得到第二个最优子树，直到得到n个最优子树的序列。

有的朋友不明白：**既然是遍历alpha，从0~+无穷，那为什么还会得到一个alpha的区间呢？**这个很好理解，因为每个alpha区间是对应一个特征节点的，而决策树的节点是有限的，因此我们真正的目的并不是遍历alpha，而是通过遍历alpha与各个g(t)比较而得到(n+1)个最优子树序列。

交叉验证

得到字数序列以后，我们可以使用独立的验证数据集，测试各子树的平方误差或基尼指数。平方误差或基尼指数最小的决策树被认为是最优的决策树。并且，每颗子树都对应着一个alpha，所以最优子树确定了，alpha也就确定了。

上面已经将CART剪枝进行详细的分析了，下面看一下CART剪枝的整个算法流程。

输入是CART树建立算法得到的原始决策树T。

输出是最优决策子树T_α。

算法过程如下：

1）初始化$\alpha_{min}=\infty$，最优子树集合$\omega=\{T\}$。

2）从叶子节点开始自下而上计算各内部节点t的训练误差损失函数$C_\alpha(T_t)$（回归树为均方差，分类树为基尼系数），叶子节点数$|T_t|$，以及正则化阈值$\alpha=min\{\frac{C(T)-C(T_t)}{|T_t|-1},\alpha_{min}\}$，更新$\alpha_{min}=\alpha$

3）得到所有节点的α值的集合M。

4）从M中选择最大的值α_k，自上而下的访问子树t的内部节点，如果$\frac{C(T)-C(T_t)}{|T_t|-1}\leq\alpha_k$时，进行剪枝。并决定叶节点t的值。如果是分类树，则是概率最高的类别，如果是回归树，则是所有样本输出的均值。这样得到α_k对应的最优子树T_k

5）最优子树集合$\omega=\omega\cup T_k$，$M=M-\{\alpha_k\}$。

6）如果M不为空，则回到步骤4。否则就已经得到了所有的可选最优子树集合ω.

7）采用交叉验证在ω选择最优子树T_α

CART算法小结

上面我们对CART算法做了一个详细的介绍，CART算法相比C4.5算法的分类方法，采用了简化的二叉树模型，同时特征选择采用了近似的基尼系数来简化计算。当然CART树最大的好处是还可以做回归模型，这个C4.5没有。下表给出了ID3，C4.5和CART的一个比较总结。希望可以帮助大家理解。

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	基尼系数，均方差	支持	支持	支持

看起来CART算法高大上，那么CART算法还有没有什么缺点呢？有，主要的缺点如下：

1）无论是ID3, C4.5还是CART,在做特征选择的时候都是选择最优的一个特征来做分类决策，但是大多数，分类决策不应该是由某一个特征决定的，而是应该由一组特征决定的。这样决策得到的决策树更加准确。这个决策树叫做多变量决策树(multi-variate decision tree)。在选择最优特征的时候，多变量决策树不是选择某一个最优特征，而是选择最优的一个特征线性组合来做决策。这个算法的代表是OC1，这里不多介绍。

2）如果样本发生一点点的改动，就会导致树结构的剧烈改变。这个可以通过集成学习里面的随机森林之类的方法解决。

决策树算法优缺点总结

我们前面介绍了决策树的特征选择，生成，和剪枝，然后对ID3, C4.5和CART算法也分别进行了详细的分析。下面我们来看看决策树算法作为一个大类别的分类回归算法的优缺点。

决策树算法的优点

1）简单直观，生成的决策树很直观。
2）基本不需要预处理，不需要提前归一化，处理缺失值。
3）使用决策树预测的代价是O(log2m)O(log2m)。 m为样本数。
4）既可以处理离散值也可以处理连续值。很多算法只是专注于离散值或者连续值。
5）可以处理多维度输出的分类问题。

6）相比于神经网络之类的黑盒分类模型，决策树在逻辑上可以得到很好的解释
7）可以交叉验证的剪枝来选择模型，从而提高泛化能力。
8） 对于异常点的容错能力好，健壮性高。

决策树算法的缺点

1）决策树算法非常容易过拟合，导致泛化能力不强。可以通过设置节点最少样本数量和限制决策树深度来改进。
2）决策树会因为样本发生一点点的改动，就会导致树结构的剧烈改变。这个可以通过集成学习之类的方法解决。
3）寻找最优的决策树是一个NP难的问题，我们一般是通过启发式方法，容易陷入局部最优。可以通过集成学习之类的方法来改善。
4）有些比较复杂的关系，决策树很难学习，比如异或。这个就没有办法了，一般这种关系可以换神经网络分类方法来解决。
5）如果某些特征的样本比例过大，生成决策树容易偏向于这些特征。这个可以通过调节样本权重来改善。

下一篇将会介绍一个决策树应用的实战内容，以及如何使用sklearn进行决策树的调参。

[1] 统计学习方法，李航，p73
[2] https://www.cnblogs.com/pinard/p/6053344.html
[3] https://blog.csdn.net/zlsjsj/article/details/81387393

推荐阅读

决策树学习笔记（二）：剪枝，ID3，C4.5
教程 | 十分钟学会函数式 Python
我们分析了633个中国城市，发现五分之二都在流失人口
决策树学习笔记（一）：特征选择