算法学习-决策树

1.决策树概念

决策树（Decision Tree）是一类常见的机器学习算法。比如，我们可以从给定的数据集学习得到一个学习模型。通过这个模型，我们可以对新添加示例进行分类。这个的一个过程可以称作"决策"或者"判定"过程。决策树是基于树结构来进行决策的。

对于一个问题进行决策时，我们常常会进行一系列的判断或者"子决策"。一般的，一颗决策树包含一个根节点，若干个内部节点和若干个子叶节点。叶节点对应决策结果，其他每个节点则对应一个属性测试。每个节点包含的样本集合根据属性测试的结果被划分到子结点中；根节点包括样本全集。从根节点到每个叶节点的路径对应了一个判定测试序列。决策树的目的时产生一棵泛化能力强，处理未见示例能力强的决策树。其基本流程遵循简单且直观的"分而治之"（Divide-and-conquer）的策略。

2.决策树的基本流程

决策树的基本流程可以用以下流程概括

输入: 训练集$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$;

属性集$A = \{a_1, a_2, \ldots, a_d\}$.

过程：函数TreeGenerate(D,A)

1:生成节点node;

2.**if** D中样本属性全属于同一类别C **then**

3: 将node标记为C类叶节点 ;**return**

4:**end if**

5:**if** A = ∅ **or** D中样本在A上取值相同 **then**

6: 将node标记为叶节点，其类别标记为D中样本最多的类；**return**

7:**end if**

8:从A中选择最优划分属性a_*;

9:**for** a_*中每一个值a_*^v **do**

10: 为node生成一个分支；令D_v表示D在a_*^v上取值的样本子集；

11: **if** D_v为空 **then**

12: 将分支节点标记为叶节点，其类别标记为D中样本最多的类；**return**

13: **else**

14: 以TreeGenerate(D_v,$A \setminus a_*$)为分支节点

15: **end if**

16:**end for**

输出：以node 为根节点的一颗决策树

有3中情况会导致递归返回，分别是：(1).当前节点包含的样本全属与一类，无法划分。(2).当前属性集为空，或是所有样本在所有属性上取值相同，无需划分。(3).当前节点包含的样本集为空，不能划分。

3.决策树的建立

3.1 划分选择

决策树学习的关键是第8行，即如何选择最优划分属性。

3.1.1信息增益

首先是信息熵的定义，信息熵是衡量信息不确定性的指标。设样本集合D 中第K 类样本所占的比例为$p_k(k=1,2,\dots|\gamma|)$,则样本 D 的信息熵定义为

$$Ent(D)=-\sum_{k=1}^{|\gamma|}p_k\log_2 p_k \tag{1}$$

$Ent(D)$ 的值越小，则 D 的纯度越高.

若离散属性 a 有 V 个取值 $\{a^1,a^2,\dots a^v\}$.我么可以使用 a 来对样本集进行划分，则可得到 V 个分支节点。第v个分支节点包含了 D 中所有属性 a 上取值为 a^v 的样本，我们记为 D^v .根据信息熵的定义可以求出D^v 的信息熵，然后根据 D^v在整个样本集中的权重，我们可以求出属性a 对样本集的信息增益。

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v) \tag{2}$$

通常情况下，信息增益越大，说明使用属性a 获得的"纯度提升越大"。这样我们可以用信息增益来进行决策树的划分属性的选择。著名的 $ID3$ 决策树学习算法就是以信息增益为准则进行划分属性选择。

3.1.2信息增益率

实际上，信息增益率会对可取值较多的属性有所偏好。为了避免这种偏好可能产生的影响，我们可以使用增益率来进行划分选择。著名的$C4.5$ 决策树算法就是采用的增益率。增益率定义为：

$$Gain_{ratio}(D,a) = \frac{Gain(D,a)}{IV(a)} \tag{3}$$

其中

$$IV(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|} \tag{4}$$

$IV(a)$ 称之为 a 的 "固有值"，若 a 属性的可取值越多，则$IV(a)$ 值通常越大。

需要注意的是增益率对属性可取值数量少的属性有所偏好。因此我们在划分选择时，先从划分属性中选择信息增益高于平均值的，然后再从中选择信息增益率高的。

3.1.3 基尼系数

$CART$ 决策树使用基尼系数来选择划分属性。数据集D 的纯度可以使用基尼值来进行度量，

$$\begin{aligned} Gini(D) &= \sum_{k=1}^{|\gamma|} \sum_{k' \neq k} p_k p'_k \\ &= 1 - \sum_{k=1}^{|\gamma|} p_k^2 \end{aligned} \tag{5}$$

$Gini(D)$ 反映了从数据集 D 中随机取两个样本，其类别标记不一致的概率，因此$Gini(D)$越小，数据纯度越高。

若将(5)式变为与(2)式相同形式，则

$$Gini_{index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) \tag{6}$$

若根据基尼系数来进行划分选择，我们可以选择使得基尼系数最小的那个属性作为最优划分选择。即：

$$a_* = \arg\min_{a \in A} \ Gini_{index}(D,a)$$

3.2 剪枝处理

剪枝处理是决策树算法为了应对过拟合的主要手段。在决策树学习过程中，有时候为了正确划分某些属性，会不断重复划分，有时候划分的分支过多，就有可能出现训练的结果"太好"，可能会把训练集中一些特点当作是所有数据的一般性质而导致“过拟合”,因此我们可以通过主动去掉一些分支，来降低过拟合的风险。

3.2.1 预剪枝

预剪枝是指在决策树生成过程中，对每个节点在划分前进行估计，若不能带来泛化能力的提升，则不进行划分。将当前节点标记为叶节点。

预剪枝可以使得决策树的很多分支没有必要展开，同时也显著降减少了决策树的生成时间和测试时间。但另一方面，虽然有些分支在当前不能带来泛化能力的提升，但是可能会在当前分支的基础上会产生带来泛化能力显著提升的分支。预剪枝基于“贪心”本质禁止这些分支展开，会导致有欠拟合的风险。

3.2.2后剪枝

后剪枝是先生成一个完整的决策树，然后自底向上的对每个节点进行判断，若将当前节点对应的子树替换成叶节点能带来泛化能力的提升，则将该子树替换为叶节点。

后剪枝通常比预剪枝保留了更多的分支。通常情况下，后剪枝欠拟合的风险很小，泛化能力往往高于预剪枝。但是后剪枝是在生成一个完整的决策树，而且在此基础上进行自底向上的对树的所有非叶节点进行考虑。因此其训练时间要比决策树和预剪枝决策耗时都要长的多。

3.3 连续值与缺失值处理

3.3.1 连续值处理

在某些情况下，数据集中的某些属性可能不再是有限个，这是我们就需要对连续值进行处理。对连续值进行处理的方法主要是连续属性离散化。这个方法正是$C4.5$ 决策树学习算法所使用的。

假定存在一个数据集 D 和连续属性 a。 a 在 D 中出现了 n 个不同的取值。将这些值按照从小到大进行排列。我们可以把排好序的值记作$\{a^1,1^2,\ldots,a^n\}$,存在划分点 t 将数据集分为 D_t^+ 和 D_t^- 。D_t^+ 为在属性 a 上取值不小于 t 的样本。D_t^- 则为在属性a上取值不大于 t 的样本。对于相邻的属性取值 a^i 和 a^{i+1} 来说，t 在区间 $[a^i,a^{i+1})$ 中取任意值划分的结果相同。因此，对于连续属性 a ,我们可以在观察在这样的一个划分集合，即

$$T_a=\left\{\frac{a^i+a^{i+1}}{2}|1\leq i\leq n-1\right\} \tag{7}$$

把区间$[a^i,a^{i+1})$ 中位点 $\frac{a^i+a^{i+1}}{2}$ 作为候选划分点，然后我们可以像考察划分点。选取最优的划分点当作样本集合的划分。

我们可以对 (2) 式进行改造，得到

$$\begin{aligned}Gain(D,a)&=\max_{t\in T_a}Gain(D,a,t)\\&=\max Ent(D)-\sum_{\lambda\in\{-,+\}}\frac{|D_t^\lambda|}{|D|}Ent(D_t^\lambda)\end{aligned} \tag{8}$$

其中，$Gain(D,a,t)$ 是在 t 点的信息增益，我们只要选择使得$Gain(D,a,t)$ 值最大的就可以当作划分点了。

同时我们需要注意，当属性划分是连续属性时，这个属性仍然可以作为子代节点的属性划分。

3.3.2 缺失值处理

缺失值处理，主要有两个问题需要解决

(1).如何在属性值缺失的情况下进行属性划分?

(2).给定划分属性，若样本在该属性上值缺失，应该如何划分?

给定训练集 D 和属性 a , $\tilde{D}$ 表示 训练集 D 的无缺失值子集。如果有缺失值，那么我们可以根据$\tilde{D}$来直接进行判断。同样，设定属性 a 有 V 个可取值 $\{a^1,a^2,\ldots,a^V\}$, 令$\tilde{D}^v$ 表示 $\tilde{D}$ 在属性a 上取值为a^V 的样本子集。$\tilde{D}_k$ 表示$\tilde{D}$ 中属于第 k 类 $(k=1,2,\ldots,k)$ 的样本子集。则有

$$\tilde{D}=\cup_{k=1}^{|\gamma|}\tilde{D}_k$$
$$\tilde{D}=\cup_{v=1}^{V}\tilde{D}^v$$

我们为每一个样本 x 赋予一个权重 w_x 并定义

$$\rho=\frac{\sum_{x\in\tilde{D}}w_x}{\sum_{x\in D}w_x} \tag{9}$$

$$\tilde{p}_k=\frac{\sum_{x\in\tilde{D}_k}w_x}{\sum_{x\in\tilde{D}}w_x} \tag{10}$$

$$\tilde{r}_v=\frac{\sum_{x\in\tilde{D}^v}w_x}{\sum_{x\in\tilde{D}}w_x} \tag{11}$$

其中对属性a, ρ 表示无缺失样本所占比例，$\tilde{p}_k$表示无缺失值样本中第 k 类所占的比例，$\tilde{r}_v$表示无缺失值样本中属性a 上取值a_v 的样本所占的比例。其中$\sum_{k=1}^{|\gamma|}\tilde{p}_k=1$, $\sum_{v=1}^{V}\tilde{r}_v=1$

基于上述定义，我们将信息增益推广为

$$\begin{aligned}Gain(D,a)&=\rho\times Gain(\tilde{D},a)\\&=\rho\times(Ent(\tilde{D}-\sum_{v=1}^{V}\tilde{r}_vEnt(\tilde{D}_v)),\end{aligned} \tag{12}$$

其中，由公式（1）,则有

$$Ent(\tilde{D})=-\sum_{k=1}^{|\gamma|}\tilde{p}_k\log_2\tilde{p}_k \tag{13}$$

对于问题（2），若样本x在属性中取值已知，则将x 划入与其取值对应的子节点，同时样本权值保持为w_x ,若样本取值未知，则将样本同时划分到所有子节点。同时权值调整为$\tilde{r}_v\cdot w_x$.简单来说就是把这个样本以不同的概率划分到不同的子节点去。

$C4.5$ 算法就是使用了上述方案。

4.Q&A

Q1.决策树常用算法。

A1：最常用$ID3$算法，$C4.5$ 算法，$CART$ 算法。

Q2.过拟合出现的情况，如何处理过拟合。

A2:

出现过拟合的情况主要是样本集中数据不够全面，或者样本中存在一些不是一般属性的属性，然后在训练过程中会把这些特征来当作一般属性进行处理。对于出现的过拟合情况，我们可以通过剪枝处理来降低过拟合的风险。根据实际情况选择是预剪枝还是后剪枝。

Q3.正则化是决策树的哪些问题。

A3:

正则化出现在决策树中的剪枝处理中，剪枝处理中会用到损失函数，

$$C_\alpha(T)=\sum_{t=1}^{|T|}|N_t|H_t(T)+\alpha|T|$$

其中经验熵

$$H_t(T)=-\sum_{k=1}^{K}\frac{|N_{tk}|}{|N_t|}log_2\frac{|N_{tk}|}{|N_t|}$$

令 $C(T)=\sum_{t=1}^{|T|}|N_t|H_t(T)$,则

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

- 剪枝就是当αα确定时，选择损失函数最小的模型，及损失函数最小的子树。
- 利用损失函数最小原则进行剪枝就是用正则化的极大似然估计进行模型选择

Q4.决策树适用范围。

A4:

（1）具有决策者期望达到的明确目标

（2）存在决策者可以选择的两个以上的可行的备选方案

（3）存在决策者无法控制的两个以上不确定因素

（4）不同方案在不同因素下的收益或损失可以计算出来

（5）决策者可以估计不确定因素发生的概率