决策树及对应算法原理

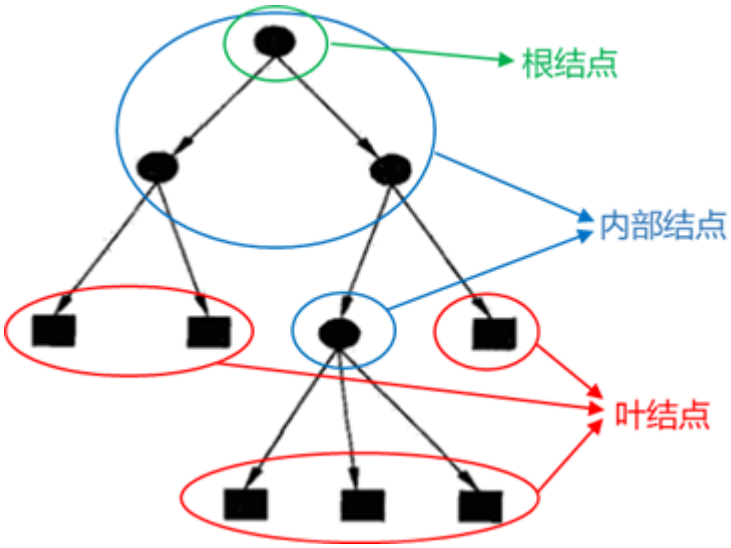
0. 参考资料

周志华 《机器学习》 清华大学出版社

李航 《统计学习方法》 清华大学出版社

1. 基本模型

一颗决策树包含一个根结点、若干个内部结点和若干个叶结点；叶结点对应于决策结果，其他每个结点则对应于一个属性测试或划分。每个结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；根结点包含样本全集。



决策树学习本质上是从训练数据集中归纳出一组分类规则。决策树的深浅（或者说叶结点的数目）控制了模型复杂度，深度为1的决策树又被称为**决策树桩（decision stump）**。

2. 特征选择

决策树学习的关键是如何选择最优划分属性。一般而言，随着划分过程不断进行，我们希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”（purity）越来越高。

2.1 信息增益

为便于说明，首先给出熵和条件熵的概念。

设X是一个取有限个值的离散随机变量，其概率分布为：

$$P(X=x_i)=p_i,\quad i=1,2,\cdots,n$$

则随机变量X的熵定义为

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

设有随机变量(X,Y)，其联合概率分布为

$$P(X=x_i,Y=y_j)=p_{ij},\quad i=1,2,\cdots,n;\quad j=1,2,\cdots,m$$

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（conditional entropy）定义为：

$$H(Y|X)=\sum_{i=1}^{n}p_iH(Y|X=x_i)$$

当熵和条件熵中的概率由数据估计（特别是极大似然估计）得到时，所对应的熵和条件熵分别称为经验熵（empirical entropy）和经验条件熵（empirical conditional entropy）。

设训练数据集为D，$|D|$表示样本个数。设有K个类C_k，$k=1,2,\cdots,K$，$|C_k|$为属于类C_k的样本个数，$\sum_{k=1}^{K}|C_k|=|D|$。设特征A有n个不同的取值$\{a_1,a_2,\cdots,a_n\}$，根据特征A的取值将D划分为n个子集$D_1,D_2,\cdots,D_n$，$|D_i|$为D_i的样本个数，$\sum_{i=1}^{n}|D_i|=|D|$。记子集D_i中属于类C_k的样本的集合为D_{ik}，即$D_k=D_i\cap C_k$，$|D_{ik}|$为D_{ik}的样本个数。

于是数据集D的经验熵$H(D)$计算如下：

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$$

特征A对数据集D的经验条件熵$H(D|A)$为：

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H\left(D_i\right)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$$

信息增益（information gain）定义为集合D的经验熵$H(D)$与特征A给定条件D的经验条件熵$H(D|A)$之差，即：

$$g(D,A)=H(D)-H(D|A)$$

一般地，熵$H(Y)$与条件熵$H(Y|X)$之差称为互信息（mutual information）。**决策树学习中的信息增益等价于训练数据集中类与特征的互信息。**

2.2 信息增益率

信息增益准则对可取值数目较多的属性有所偏好，为减少这种偏好可能带来的不利影响，可以使用**增益率（gain ratio）**来选择最优划分属性。

增益率定义为：

$$g_R(D,A)=\frac{g(D,A)}{\mathrm{IV}(A)}$$

其中，

$$\mathrm{IV}(A)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\log_2\frac{|D_i|}{|D|}$$

称为属性A的固有值（intrinsic value）。特征A的可能取值数目越多，则$\mathrm{IV}(A)$的值通常会越大。不过需注意的是，增益率准则对可取值数目较少的属性有所偏好。

3. 决策树的生成

本节介绍经典的决策树生成算法：ID3算法和C4.5算法，这些都是通过**递归**的方式生成决策树。

3.1 ID3算法

ID3算法使用信息增益作为划分准则，算法如下：

输入：训练数据集D，特征集A，阈值ε；

输出：决策树T

1：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作为该结点的类标记，返回T；

2：若$A=\varnothing$，则T为单结点树，并将D中实例数最多的类C_k作为该结点的类标记，返回T；

3：否则，计算A中各特征对D的信息增益，选择信息增益**最大**的特征A_g；

4：如果A_g的信息增益小于阈值ε，则置T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T；

5：否则，对A_g的每一可能值a_i，依$A_{\mathrm{g}}=a_i$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，有结点及其子结点构成树T，返回T；

6：对第i个子结点，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用步骤1~步骤5，得到子树T_i，返回T_i。

3.2 C4.5算法

C4.5算法使用增益率作为划分准则，其它与ID3算法类似，具体如下：

输入：训练数据集D，特征集A，阈值ε；

输出：决策树T

1：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作为该结点的类标记，返回T；

2：若$A=\varnothing$，则T为单结点树，并将D中实例数最多的类C_k作为该结点的类标记，返回T；

3：否则，计算A中各特征对D的信息增益，选择信息增益率**最大**的特征A_g；

4：如果A_g的信息增益率小于阈值ε，则置T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T；

5：否则，对A_g的每一可能值a_i，依$A_{\mathrm{g}}=a_i$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，有结点及其子结点构成树T，返回T；

6：对第i个子结点，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用步骤1~步骤5，得到子树T_i，返回T_i。

4. 决策树的剪枝

决策树的生成算法一般都会产生一个过拟合的模型，这是可以使用“剪枝”（pruning）的手段来去掉一些分支，从而降低过拟合风险。具体地，剪枝从已生成的树上裁掉一些子树或叶结点，并将其根结点或父结点作为新的叶结点，从而简化分类模型。

《统计学习原理》一书中介绍了一种简单的剪枝算法。决策树的剪枝往往通过极小化决策树整体的损失函数来实现。设树T的叶结点个数为$|T|$，t是树T的叶结点，该叶结点有N_t个样本点，其中k类的样本点有N_{tk}个，$k=1,2,\cdots,K$，$H_t(T)$为叶结点t上的经验熵，$\alpha\geqslant 0$为参数，则决策树学习的损失函数可以定义为：

$$C_\alpha(T)=\sum_{t=1}^{|T|}N_tH_t(T)+\alpha|T| \tag{4.1}$$

其中经验熵为

$$H_t(T) = -\sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}$$

我们可以将式（4.1）右端的第一项记为

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

这时有

$$C_\alpha(T) = C(T) + \alpha|T| \tag{4.2}$$

式（4.2）中，$C(T)$表示模型对训练数据的预测误差，$|T|$表示模型复杂度，参数$\alpha \geqslant 0$控制两者之间的影响。

于是，我们可以给出剪枝的算法，如下：

输入：生成算法产生的整个树T，参数α

输出：修剪后的子树T_α

1：计算每个结点的经验熵

2：递归地从树的叶结点向上回缩

设一组叶结点回缩到其父结点之前与之后的整体树分别为T_B与T_A，其对应的损失函数分别是$C_\alpha\left(T_B\right)$与$C_\alpha\left(T_A\right)$，如果

$$C_a\left(T_A\right) \leqslant C_\alpha\left(T_B\right)$$

则进行剪枝，即将父结点变为新的叶结点。

3：返回步骤2，直至不能继续为止，得到损失函数最小的子树T_α。

5. CART算法

CART算法的全称是分类与回归树（classification and regression tree，CART）。**CART假设决策树是二叉树**，内部结点特征的取值为“是”或“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。

5.1 回归树生成

一个回归树对应着特征空间的一个划分以及在划分的单元上的输出值。假设已将输入空间划分为M个单元$R_1, R_2, \cdots, R_M$，并且在每个单元R_m上有一个固定的输出值c_m，于是回归树模型可表示为

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

当输入空间的划分确定时，可以用平方误差$\sum_{x_i \in R_n}\left(y_i - f\left(x_i\right)\right)^2$来表示回归树对于训练数据的预测误差。易知，单元$R_m$上的$C_m$的最优值$\hat{c}_m$是$R_m$上的所有输入实例$x_i$对应的输出$y_i$的均值，即

$$\hat{c}_m = \operatorname{avg}(y_i | x_i \in R_m)$$

剩下的问题就是怎么对输入空间进行划分。这里采用启发式算法，选择第j个变量$x^{(j)}$和它的取值s，作为切分变量和切分点，并定义两个区域：

$$R_1(j, s) = \{x | x^{(j)} \leqslant s\} \quad \& \quad R_2(j, s) = \{x | x^{(j)} > s\}$$

然后寻找最优切分变量j和最优切分点s。具体地，求解

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

对固定输入变量j可以找到最优切分点s。

$$\hat{c}_1=\text{avg}(y_i|x_i\in R_1(j,s))\quad\&\quad\hat{c}_2=\text{avg}(y_i|x_i\in R_2(j,s))$$

遍历所有输入变量，找到最优的切分变量j，构成一个对(j,s)。依此将输入空间划分为两个区域。接着，对每个子区域重复上述过程，直到满足停止条件为止，这样就生成了一颗最小二乘回归树。

5.2 分类树的生成

分类树用**基尼指数（Gini index）**选择最优特征，同时决定该特征的最优二值切分点。

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

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

对于给定的样本集合D，其基尼系数为

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

这里，C_k是D中属于第k类的样本子集，K是类的个数。

如果样本集合D个根据特征A是否取某一可能值a被分割为D_1和D_2两部分，即

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

则在特征A的条件下，集合D的基尼指数定义为

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

基尼指数$\text{Gini}(D)$表示集合D的不确定性，基尼指数$\text{Gini}(D,A)$表示经$A=a$分割后集合D的不确定性。基尼指数越大，样本集合的不确定性也就越大。

在划分时，对于每个特征A和它们的每个可能取值a都计算基尼指数，选择基尼指数最小的特征A对应的a。

一般来说，算法的停止条件有三：1）结点中样本个数小于某个阈值；2）样本集的基尼指数小于某个阈值（样本基本属于同一类）；3）没有更多特征。

5.3 CART剪枝

CART剪枝算法由两步组成：首先从生成算法产生的决策树T_0底端开始不断剪枝，直到T_0的根结点，形成一个子树序列$\{T_0,T_1,\cdots,T_n\}$；然后通过交叉验证法在独立的验证数据集上对子树序列进行测试，从中选出最优子树。

在剪枝的过程中，子树的损失函数为：

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

其中，T为任意子树，$C(T)$为对训练数据的预测误差（如基尼指数），$|T|$为子树中的叶结点个数，$\alpha\geqslant 0$为参数，$C_\alpha(T)$为参数是α时的子树T的整体损失。参数a权衡训练数据的拟合程度与模型的复杂度。

具体地，从整体树T_0开始剪枝。对T_0的任意内部结点t，以t为单结点树的损失函数是

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

以t为根结点的子树T_t的损失函数是

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

当$\alpha = 0$及α充分小时，由不等式

$$C_\alpha\left(T_t\right) < C_\alpha(t)$$

当α增大时，在某一α有

$$C_\alpha\left(T_t\right) = C_\alpha(t)$$

此时，有

$$\alpha = \frac{C(t) - C\left(T_t\right)}{\left|T_t\right| - 1}$$

并且T_t与t有相同的损失函数值，而t的结点更少，因此t比T_t更可取，对T_t进行剪枝。

为此，对T_0中每一内部结点t，计算

$$g(t) = \frac{C(t) - C\left(T_t\right)}{\left|T_t\right| - 1}$$

它表示剪枝后整体损失函数减少的程度。

于是，CART剪枝算法如下：

输入：CART算法生成的决策树T_0；

输出：最优决策树T_α

1：设$k = 0$，$T = T_0$

2：设$\alpha = +\infty$

3：自上而下地对各内部结点t计算$C\left(T_t\right)$，$\left|T_t\right|$以及

$$g(t) = \frac{C(t) - C\left(T_t\right)}{\left|T_t\right| - 1}$$

$$\alpha = \min(\alpha, g(t))$$

这里，T_t表示以t为根结点的子树，$C\left(T_t\right)$是对训练数据的预测误差，$\left|T_t\right|$是T_t的叶结点个数。

4：自上而下地访问内部结点t，如果有$g(t) = \alpha$，进行剪枝，并对叶结点t以多数表决法决定其类，得到树T。

5：设$k = k + 1$，$\alpha_k = \alpha$，$T_k = T$。

6：如果T不是由根结点单独构成的树，则回到步骤4。

7：采用交叉验证法在子树序列$T_0, T_1, \cdots, T_n$中选取最优子树T_α。