问题背景

假设给定训练数据集 D，其样本容量为 N：

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

其中，x_i是 n 维的输入特征向量，$x_i = \left(x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)}\right)^T$，类标记$y_i \in \{1,2,\dots,K\}$。

问题设定

根据训练数据集构建决策树模型，使它可以对实例正确分类。并且，决策树应足够准确和足够简单（或者说，应具有较小的经验误差和结构误差）。

问题求解

最优决策树的求解是 *NP* 问题

对于同样的训练数据集，存在多种构造方式可以使决策树正确分类训练集中的实例。即存在着一个决策树集合，集合中的任一决策树均可以正确分类实例。但是，在决策树集合中也存在着较优解，有的决策树在泛化过程中有着更好的表现。最优决策树的寻找是 *NP* 完全问题，没有显式方法可以用于寻找最优决策树，只能不断生成和对比，通过启发式方法获得令人满意的决策树解。

最优决策树具有极小的经验误差和结构误差，将最优决策树的寻找过程分解为生成和剪枝两个部分，前者用于实现极小的经验误差，后者用于控制结构误差。

启发式的树生成方法

开始时将所有训练数据放置于根节点，接下来，需要寻找到一个“最优特征”将训练数据集向下分割为多个子集。

思考我们的初衷，我们希望使用最简单的决策树结构，去实现经验误差最小的分类过程。在最理想的情况下，经过从根节点出发的第一次分割后，有着同样类标记的样本均被归结到同一子集，且有着不同类标记的样本分处不同的子集。

也就是说，我们希望决策树在每次执行向下分割动作之后，能够尽可能地将有着同样类标记的样本分类到同一个子集中去。因此，数据集的分割问题得到转化：根据何种准则选择特征用于执行分割动作，可以实现前述的分类效果？或者说，如果比较样本中各个特征间的该种属性？

熵与不确定度

已知样本容量为 *N* 的训练数据集 *D*，样本有 *K* 个种类，类标记$y_i \in \{1,2,\dots,K\}$。熵可以衡量样本分布的多样性：

$$H(D) = -\Sigma_{i=1}^{K} p_i \log p_i \ , \quad \text{式中} p_i = \frac{|D_i|}{|D|}$$

式中，$|D_i|$表示训练集中类标记为 *i* 的样本个数，$|D|$表示训练集中的样本个数，$|D|$=*N*。

熵取值的范围是[0, 1]。熵值越大，说明数据集全体样本在 *K* 个种类中的分布越均匀，即越多样；熵值越小，说明数据集全体样本分布越失衡，即越确定；熵值为 0，说明全体样本均属于同一类别。

熵表征着分布的多样性，它只和样本的分布有关，与样本标记的取值无关。所以 *H(D)* 常写作 *H(p)*：

$$H(p) = H(D) = -\Sigma_{i=1}^{K} p_i \log p_i$$

特征引入、条件熵和信息增益（1）

由于特征 *A* 的引入，以此将数据集 *D* 划分为了 *m* 个子集:

$$D = \{D_{m,1}, D_{m,2}, \dots, D_{m,m}\}$$

引入条件熵的概念，用于表征使用特征 *A* 对数据集 *D* 进行划分后，划分结果的多样性：

$$H(D|A) = \sum_{i=1}^{m} p_{m,i} H(D_{m,i}) = \sum_{i=1}^{m} \frac{|D_{m,i}|}{|D|} \left(-\sum_{j=1}^{K} \frac{|D_{m,i}(y=j)|}{|D_{m,i}|} log \frac{|D_{m,i}(y=j)|}{|D_{m,i}|} \right)$$

$$= -\sum_{i=1}^{m} \frac{|D_{m,i}|}{|D|} \sum_{j=1}^{K} \frac{|D_{m,i}(y=j)|}{|D_{m,i}|} log \frac{|D_{m,i}(y=j)|}{|D_{m,i}|}$$

条件熵表示使用特征 *A* 对数据集 *D* 进行划分后，得到的划分子集中样本类标记分布的多样性。条件熵的取值范围也是[0, 1]。

特征 *A* 的引入，使得数据集的分布情况发生了变化。引入特征 *A* 之前，数据集分布的多样性是 *H(p)*，而引入特征 *A* 之后，由各个数据子集中类标记的分布多样性共同组成的数据集分布多样性是 *H(D|A)*。

如果特征 *A* 具有很强大的将有着同样类标记的样本分类到同一个子集中去的能力，那么在引入特征 *A* 对数据集进行分类后，数据集中类标记的分布多样性 *H(D|A)*应远远小于引入特征 *A* 之前的数据集中的类分布多样性 *H(p)*。

特征引入、条件熵和信息增益（2）

类分布多样性减少的原因是由于特征 A 的引入向数据集分布中注入了信息。由特征 A 的引入而产生的信息增益定义为 *g*(*D*, *A*)：

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

如果特征 *B* 注入的信息增益大于特征 *A*，即：

$$g(D, B) > g(D, A)$$

则引入特征 *B* 用于划分数据集优于引入特征 *A*。

信息增益与信息增益比

使用信息增益作为选择特征的依据，将倾向于选择特征取值较多的特征。则决策树在同一深度上将有更多的孩子节点，这将增大决策树的广度复杂度。信息增益比可以校正这一现象，将决策树的生成朝着增大深度复杂度的方向发展。

由于特征 A 的引入，产生的信息增益是 $g(D, A) = H(D) - H(D|A)$。同时，由于使用特征 A 对数据集进行划分，使得数据集按照 m 个数据子集的形式重新分布。生成数据子集的同时也生成了新的数据子集分布熵$H_A(D)$：

$$H_A(D) = -\sum_{i=1}^{m} \frac{|D_i|}{|D|} log \frac{|D_i|}{|D|}$$

考虑引入特征 A 带来的产出和投入，定义信息增益比$g_R(D,A)$：

$$g_R(D,A) = \frac{产出}{投入} = \frac{g(D,A)}{H_A(D)}$$

如果引入特征 B 带来的信息增益比大于特征 A，即：

$$g_R(D,B) > g_R(D,A)$$

则在增加决策树的深度复杂度的背景下，引入特征 B 用于划分数据集优于引入特征 A。

决策树生成：基于信息增益或信息增益比的特征选择过程

ID3 算法（基于信息增益生成决策树）

输入：训练数据集 D，特征集 A，阈值ε；

输出：决策树 T。

1. （判收敛）如果 D 中所有实例属于同一类 C_k，则 T 为单节点树，将 C_k 作为该节点的类标记，返回 T；
2. （判空）如果 $A=\emptyset$，则T 为单节点树，将 D 中实例数最大的类 C_k 作为该节点的类标记，返回 T；
3. 计算特征集 A 中各个特征对数据集 D 的信息增益，选择信息增益最大的特征 A_g；
4. 如果的信息增益小于阈值，则置 T 为单节点树，将 D 中实例数最大的类 C_k 作为该节点的类标记，返回 T；
5. 否则，根据 A_g 的每个可能值将数据集 D 划分为若干非空子集 D_i，将 D_i 中实例数最大的类作为标记，构建子节点，由节点及其子节点构成树 T，返回 T；
6. 对于第 i 个子节点，以 D_i 为训练集，以 A-A_g 为特征集，递归调用第 1~5 步，得到子树 T_i，返回子树 T_i。

C4.5 算法（基于信息增益比生成决策树）

输入：训练数据集 D，特征集 A，阈值ε；

输出：决策树 T。

1. （判收敛）如果 D 中所有实例属于同一类 C_k，则 T 为单节点树，将 C_k 作为该节点的类标记，返回 T；
2. （判空）如果 $A=\emptyset$，则T 为单节点树，将 D 中实例数最大的类 C_k 作为该节点的类标记，返回 T；
3. 计算特征集 A 中各个特征对数据集 D 的信息增益比，选择信息增益比最大的特征 A_g；
4. 如果的信息增益比小于阈值，则置 T 为单节点树，将 D 中实例数最大的类 C_k 作为该节点的类标记，返回 T；
5. 否则，根据 A_g 的每个可能值将数据集 D 划分为若干非空子集 D_i，将 D_i 中实例数最大的类作为标记，构建子节点，由节点及其子节点构成树 T，返回 T；
6. 对于第 i 个子节点，以 D_i 为训练集，以 A-A_g 为特征集，递归调用第 1~5 步，得到子树 T_i，返回子树 T_i。

决策树的剪枝

如前所述，决策树仅依据经验误差最小化完成树的生成。剪枝则是在考虑经验误差的基础上，减少决策树的复杂程度。即在经验误差的基础上引入正则项构造成本函数，求解成本函数取极值时的决策树配置。

如何度量一棵决策树的经验误差？叶子节点上样本集合的分布多样性可以表征树的经验误差，叶子节点上的样本集合分布越多样，经验误差越大，样本分布的熵也就越大。

设树 T 的叶节点个数为$|T|$，N_t 表示叶节点 t 上的样本数量，$N_t(y=i)$表示在这 N_t 个样本中，类标记为 i 的样本的数量。下式表示了决策树的经验误差：

$$C(T)=\sum_{t=1}^{|T|}N_tH_t(T)=-\sum_{t=1}^{|T|}N_t\sum_{i=1}^{K}\frac{N_t(y=i)}{N_t}log\frac{N_t(y=i)}{N_t}$$

$$=-\sum_{t=1}^{|T|}\sum_{i=1}^{K}N_t(y=i)\,log\frac{N_t(y=i)}{N_t}$$

如何度量一棵决策树的复杂度？树的叶子节点的个数可以表征树的复杂度，叶子节点的数量越多，说明决策树最终的分支越多，决策树的结构也就越复杂。下式表示了决策树的结构误差，其中α是人工设定的调节因子，如果设定为较小值，将促使成本函数取极小值时生成复杂度较高的决策树，如果设定为较大值，将生成复杂度较低的决策树：

$$\alpha|T|$$

综合经验误差和结构误差，得到剪枝决策树时的成本函数：

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

使用成本函数，即可以从树的根节点往上进行剪枝。

树的剪枝算法：

输入：生成算法生成的整棵树 T，参数α；

输出：成本函数极小的决策树T_α。

1. 计算每个节点的经验熵，即 $H_t(T)$；
2. 递归地从树的叶节点向上回缩：

   设一组叶节点回缩到其父节点之前与之后的整体树分别为 T_B 与 T_A，分别对应的成本函数数值是$C_\alpha(T_B)$和$C_\alpha(T_A)$，如果$C_\alpha(T_A) \le C_\alpha(T_B)$，则进行剪枝，即父节点变为新的叶子节点；
3. 重复第 2 步，直至遍历完整棵树，得到成本函数最小的决策树T_α。

对于处在同一深度但前驱节点不同的叶子节点，可以动态规划地进行第 2 步，即独立地分别进行。

基尼指数和 CART 二叉树生成算法

CART 假设决策树是二叉树。它既可以用于分类，也可以用于回归。

最小二乘回归树生成算法：

输入：训练数据集 D；

输出：回归树 $f(x)$。

1. 使用平方误差最小化作为成本函数，用于求解最优的划分变量 j 和划分点 s：

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

   式中，划分点 s 将特征空间划分成 R_1 和 R_2 两个区域，c_1 和 c_2 是两个区域中样本函数值的平均值。遍历变量 j，对固定的划分变量 j 扫描划分点 s，选择使平方误差最小的对(j, s)。
2. 使用选定的对(j, s)划分区域并使用 c_1 和 c_2 作为两个区域上的输出值；
3. 继续对两个区域执行第 1~2 步，直至满足停止条件
4. 得到决策树。

分类树的生成算法：

输入：训练数据集 D，停止计算的条件（可以是节点中的样本个数小于预定阈值；或者样本集的基尼指数小于预定阈值，即样本基本属于同一类；或者没有更多特征）；

输出：*CART* 决策树。

1. 对于训练数据集 D，根据现有的每个特征对其每个可能的取值的基尼指数；

   对于给定的训练数据集 D，假设数据集 D 中有 K 个类别，C_k 表示中属于第 k 类的

样本子集，则数据集的基尼指数为：

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

在特征 *A* 及其取值 *a* 的条件下，数据集 *D* 的基尼指数定义为：

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

式中，$D_1 = \{(x, y) \in D \mid A(x) = a\}, D_2 = D - D_1$，$Gini(D_1) = 1 - \sum_{k=1}^{K} \left(\frac{|D_1(y \in C_k)|}{|D_1|}\right)^2$

2. 选择基尼指数最小的特征及其对应的划分点作为最优特征和最优划分点，根据最优特征和最优划分点，从现节点生成两个子节点，将训练数据集按照特征分配到两个子节点中；
3. 对得到的两个子节点依次调用第 1~2 步，直至满足停止条件；
4. 得到决策树。

CART 二叉树剪枝

嵌套生成子树序列，再交叉验证挑选最优子树。

在树 *T* 中，对于任意内部节点 *t*，以其为单节点树的成本函数是：

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

以其为根节点的子树 T_t 的成本函数是：

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

当$\alpha = 0$或充分小时，有不等式：

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

当α增大时，在某一α有：

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

当α再增大时：

$$C_\alpha(T_t) > C_\alpha(t)$$

当$C_\alpha(T_t) = C_\alpha(t)$时，即$\alpha = \frac{C(t) - C(T_t)}{|T_t| - 1}$时，$T_t$ 和 *t* 有相同的成本函数，而 *t* 的节点更少，为了降低结构误差，故对 T_t 进行剪枝。

α有很多种取值，可以自下而上地遍历树的内部节点以生成不同的α，即得到升序的α序列，每个α对应着一棵子树。然后再使用独立于训练数据的验证数据集，根据平方误差最小或基尼指数最小，在子树序列中交叉验证，挑选最优子树作为剪枝后的 *CART* 二叉树输出。

CART 剪枝算法：

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α=+∞；

3. 自下而上地对树的内部节点 *t* 计算$C(T_t)$，$|T_t|$以及

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

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

4. 对$g(t) = \alpha$的内部节点进行剪枝，并对叶子节点 *t* 以多数表决法决定其类别，得到树 *T*；
5. 设 k=k+1，$\alpha_k = \alpha$，T_k=*T*；
6. 如果 T_k 不是由根节点及两个叶子节点构成的树，则执行第 3~5 步；否则另 $T_k = T_n$；
7. 使用独立于训练数据的验证数据集，根据平方误差最小或基尼指数最小，在子树序列中交叉验证，挑选最优子树作为剪枝后的 *CART* 二叉树输出。