分类决策树(Decision Tree)

- 模型
  - 划分特征空间，利用树形结构进行分类(预测类别)或回归(预测数值)，内部每个节点代表一个属性或特征，叶子结点表示一个类别
  - 两种表述
    - 1、规则集合：构建一组if-then规则集合，从根节点到叶子结点是一条路径，路径间满足互斥完备。
    - 2、条件概率：将特征空间划分为互不相交的单元，在各单元内定义类的概率分布，每一条路径即为一个条件。各节点以条件概率最大一类作为预测结果。
  - 是一种判别模型，属于符号主义，可解释性强
- 策略
  - 选择一组规则或构建一个概率分布，使得该规则或条件概率分布能够拟合训练样本，并具有很好的泛化能力
  - 风险函数：基于正则化的极大似然估计
    - $C_\alpha(T) = C(T) + \alpha|T|$
      - 右一为经验风险
      - 右二为正则化项
      - T表示叶子节点，α表示惩罚系数，提高α可降低树的深度
- 算法
  - 相关概念
    - 经验熵 $H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log \frac{|C_k|}{|D|}$
    - 条件熵 $H(D|A) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} log \frac{|D_{ik}|}{|D_i|}$
    - 信息增益 $g(D, A) = H(D) - H(D|A)$
    - 信息增益比 $g_R(D, A) = g(D, A)/H_A(D)$
    - 基尼指数 $Gini(D) = 1 - \sum_{k=1}^{K} (\frac{|C_k|}{|D|})^2$
    - 条件基尼指数 $Gini(D, A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} (1 - \sum_{k=1}^{K} (\frac{|D_{ik}|}{|D_i|})^2)$
  - 特征选择
    - 启发式算法，局部寻优，选择分类效果最好(信息增益、信息增益比、基尼系数)的特征或属性作为内部节点
  - 特征连续值处理
    - 针对连续值特征，采用二分法对特征值进行切分，选择切分后信息增益最大的作为切分点
  - 决策树生成
    - ID3算法-分类树
      - 1、从根节点开始，计算各个特征的信息增益，以信息增益最大的作为节点属性
      - 2、根据节点属性值，划分为若干非空集合，对于每个集合，从剩余特征中选择信息增益最大的作为节点属性
      - 3、以此类推，直到属性为空或单一类别或信息增益小于设定阈值(预剪枝)则停止，得到叶子节点
    - C4.5算法-分类树
      - 与ID3算法唯一不同的是：采用信息增益比来求解
    - CART算法-分类二叉树
      - 1、从根节点开始，二分各个特征并计算其条件基尼指数，取基尼指数最小的作为节点属性，并根据该属性的最优切分点将样本分为两类。
      - 2、在剩余特征中，用同样的方法对所分样本集再次进行二分类
      - 3、当基尼指数小于设定阈值(预剪枝)或属性为空或单一类别时则停止，得到叶子节点
  - 剪枝
    - 目的：优化整体模型，降低过拟合风险
    - 预剪枝
      - 在生成决策树的过程剪枝，如设定深度、增益阈值、节点样本数等
    - 后剪枝
      - 一种固定自底而上固定α方法
        - $C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha|T|$
          - T表示叶子节点，N表示叶子结点样本数，H表示叶子节点熵，α表示惩罚系数
        - 1、给定惩罚系数α
        - 2、计算每个节点的经验熵
        - 3、从叶子结点向上修剪
        - 4、若修剪后的风险降低则剪掉该叶节点，修剪后父节点转为叶节点，并以其中多数作为当前类别
        - 5、以此类推，直到不能继续为止
      - CART剪枝
        - 1、α=+∞
        - 2、计算内部节点t
          - 单节点t的损失 $C_\alpha(t) = C(t) + \alpha$
          - 以t为节点树的损失 $C_\alpha(T) = C(T) + \alpha|T|$
          - 计算节点t的剪枝临界值g(t) $g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1}$
        - 3、根据g(t)从小到大依次剪枝，得到子树序列{T1,...Tn}，并依次将g(t)值设为对应子树的α即{α1,...,αn}。规定，剪枝后的叶节点以其多数类作为其分类。
        - 4、采用交叉验证法选择最优的子树
- API
  - sklearn.tree.DecisionTreeClassifier()
- 主要应用
  - 分类
    - 可解释性强，非常适合于高风险决策领域
  - 回归
    - 相对于分类，应用不多
- 优劣分析
  - 优点
    - 简单直观，逻辑性强，容易解释
    - 可不需要预处理，如缺失值、归一化等
    - 一般对数据类型不做要求，可同时处理类别型和数值型数据
    - 效率高，决策树构建后，每次预测的计算次数不超过最大深度
  - 缺点
    - 容易过拟合
    - 为一个NPC问题，采用启发式算法，但易陷入局部最优
    - 容易因为样本的轻微变动而导致决策树结构发生剧烈变化
    - 每次分类只考虑了一个特征，但实际上一个分类可能跟多个特征相关

浮动主题