# 随机森林

七月算法

# 主要内容

- 决策树
  - 生成与裁剪算法
- 集成学习－Bagging思路
  - 随机森林
- RF的代码实现与数据实验

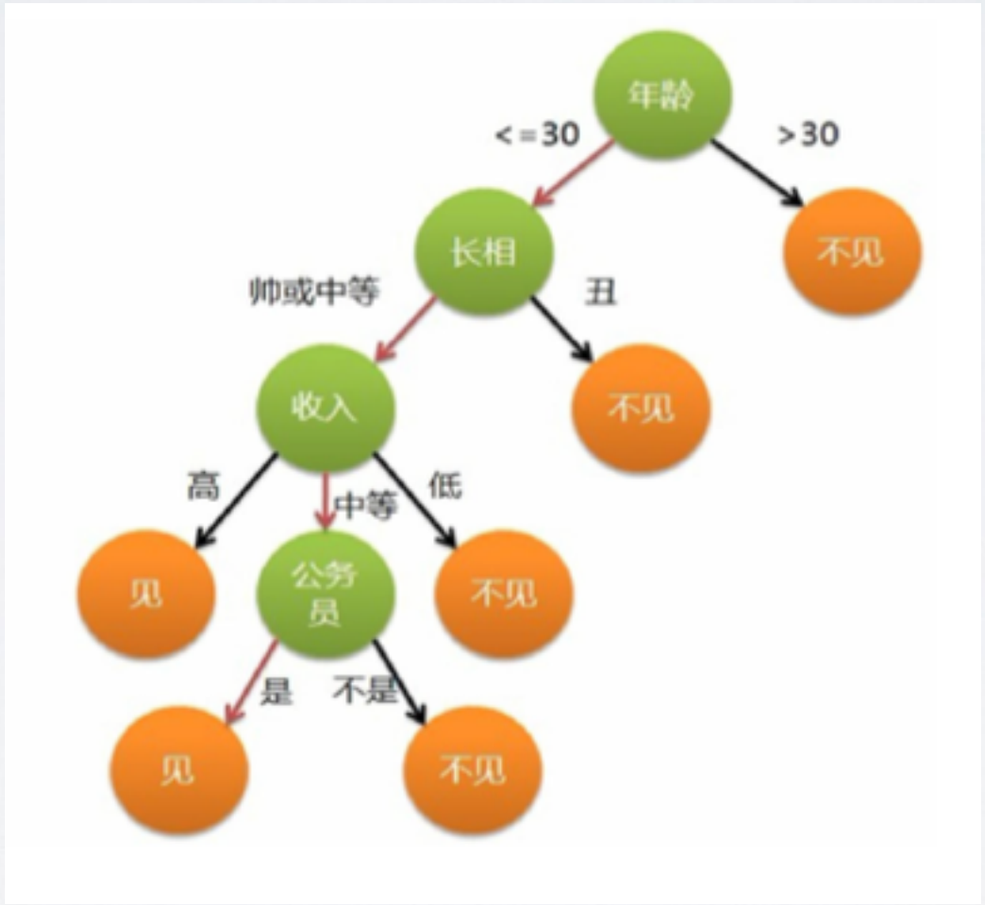
# 决策树定义

- 决策树的定义：
  - 每个非叶节点表示一种对样本的分割，通常是选用样本的某一个特征，将样本分散到不同子节点中
  - 子节点继续对分散来的样本继续进行分割操作
  - 叶子节点表示输出，每个分散该叶节点中样本都属于同一类（或近似的回归值）

# 决策树架构

- 决策树学习：
  - 一种根据样本为基础的归纳学习
  - 采用的是自顶向下的递归方法：开始数据都在根节点，递归的进行数据分片
  - 通过剪枝的方法，防止过拟合
- 决策树的使用：
  - 对未知数据进行分类
  - 按照决策树上生成时所采用的分割属性逐层往下，直到一个叶子节点

# 决策树的示意



# 树生成方法

- 如何实现分割?
    - 选择一个特征，设置一个阈值（threshold），Decision Stump
- 实现什么样的分割?
    - 分类效果最好（或分类最纯的，或能使树的路径最短）
    - 度量方法
        - 信息增益（ID3）
        - 信息增益率（C4.5）
        - 基尼指数（CART）

# 信息增益/率

- 某个节点分割前的熵值：经验熵（empirical entropy）

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

- 分割后的熵值：经验条件熵

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

- 所谓的信息增益/率

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

$$g_r(D,A)=g(D,A)/H(A)$$

  - 我们选择信息增益最大的那个分割

# 基尼指数

- 计算方法

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

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

- 分割后的基尼指数计算，对每个子节点加权求和

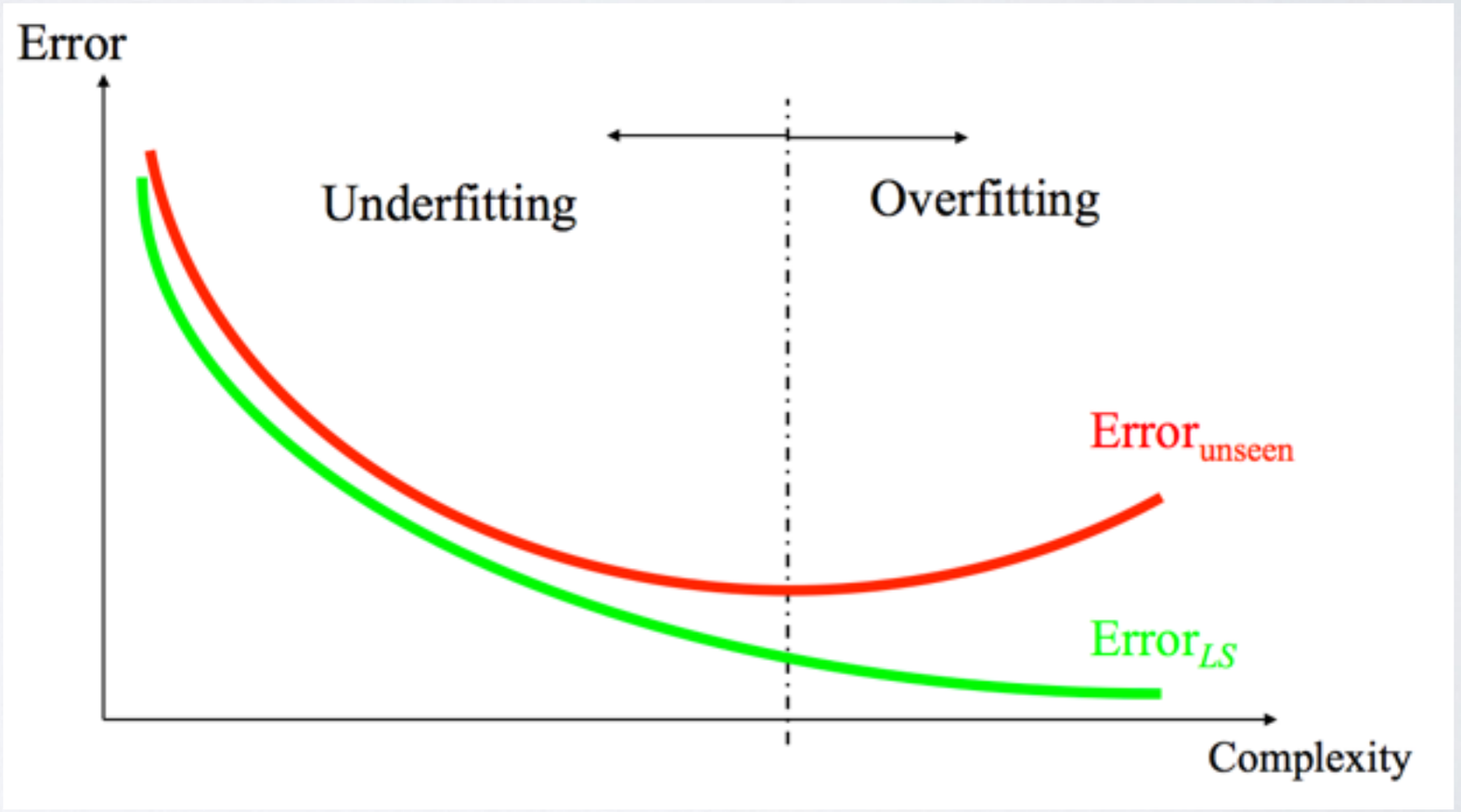
$$Gini(D|A)=\sum\frac{D_i}{D}Gini(D_i)$$

# 生成方法总结

- 执行一个分割的信息增益越大，表明这个分割对样本的熵减少的能力越强，这个分割所在的特征使得数据由不确定性变成确定性的能力强。

# 过拟合

- 一个对样本完全分类的树（完整树），容易过拟合，泛化能力较弱
  - 样本噪声
  - 样本量少



# 树的剪枝

- 设计一个树的分类误差评价函数，对所有叶节点的熵加权求和，即：

$$C(T)=\sum_{t\in leaf} N_t \cdot H(t)$$

- 可以使用叶节点数目复杂度评估函数，剪枝的目标就是在这两个函数之间作出平衡，设计出最后的评价函数

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

- 因此树的剪枝，就是挑选出完整树的子树，使得评价函数值最小

# 树剪枝算法

- 第一种方法，固定某个经验值 $\alpha$，生成唯一的使得评价函数最小的树
- 第二种方法，通过迭代操作，构造一系列不同 $\alpha$ 值，但是评价函数接近的备选树，然后通过交叉验证的方法选择最好的树

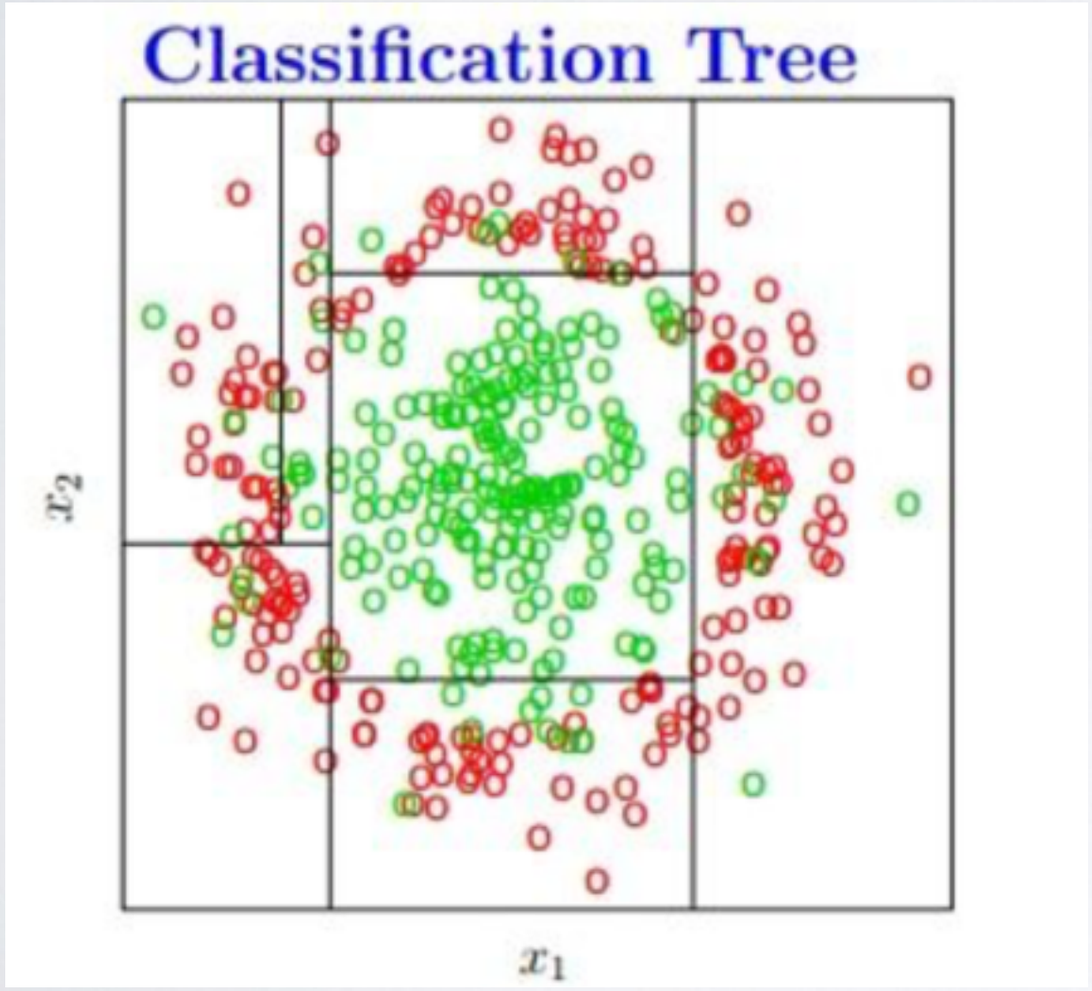
# 如何构造备选树

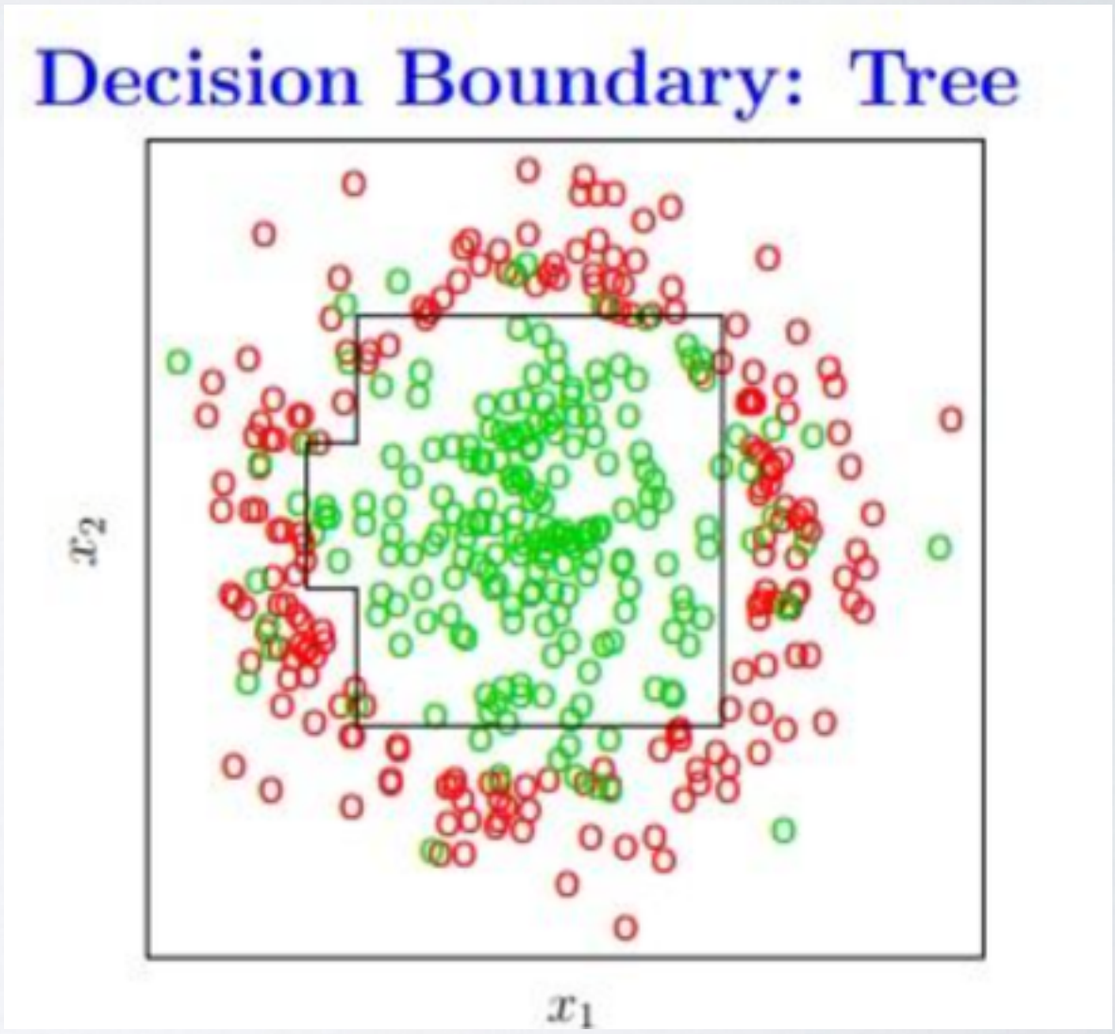
- 从完整树开始，当$\alpha$为0的时候，明显完整树是最优树，即第一个备选树，评价函数值就是 $C(T_0)$
- 现在裁剪一个r节点，其他部分不动，因此我们可以独立考察r节点构成的单节点树和根节点树的评价函数的变化

$$C_\alpha(t_r) = C(t_r)+\alpha \qquad C_\alpha(T_r) = C(T_r)+\alpha|T_r|$$

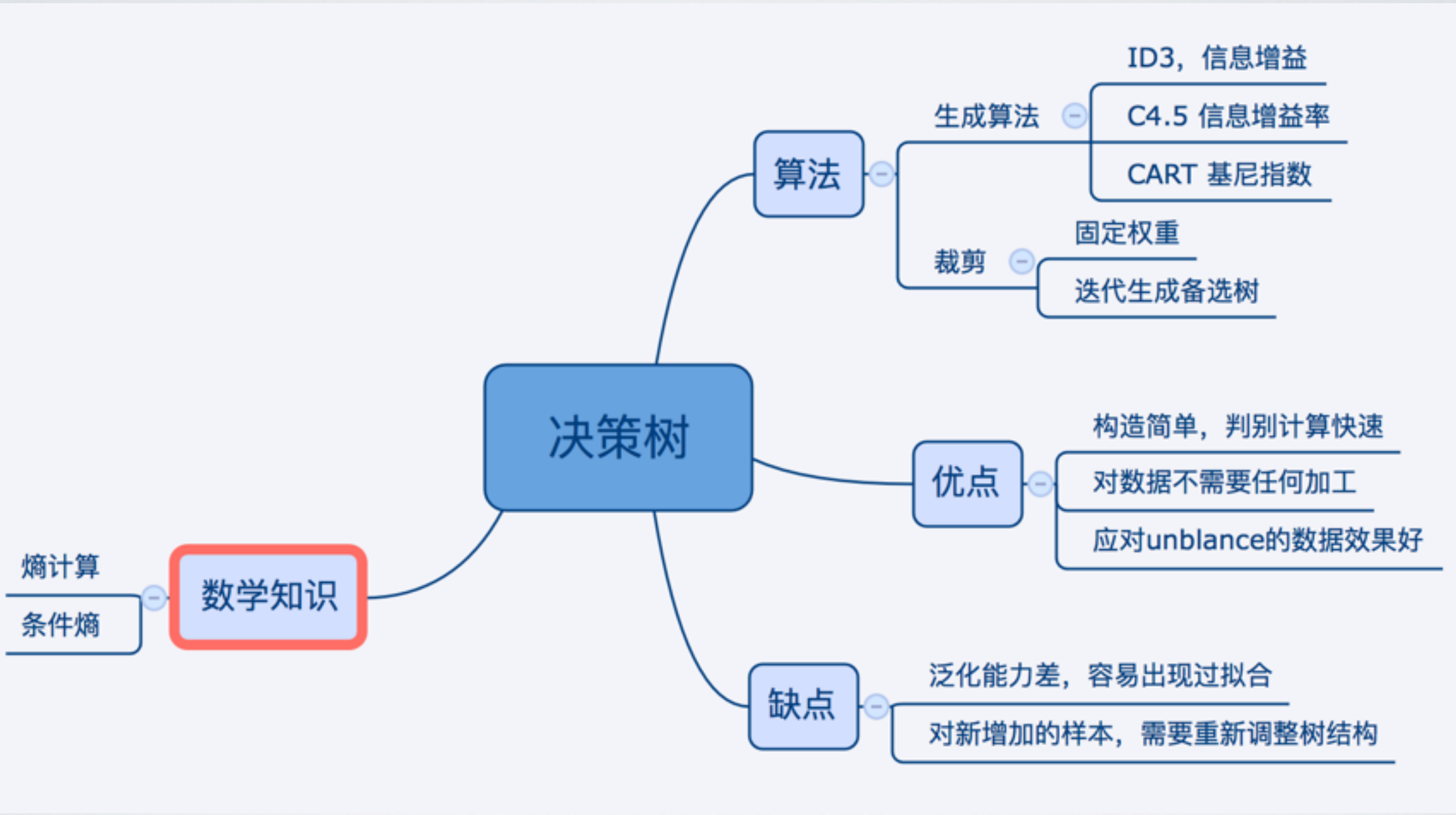
- 当 $\alpha$ 比较小的时候, $C_\alpha(T_r) < C_\alpha(t_r)$，可以慢慢增大$\alpha$，使得不等式变为等式，为了使得 $\alpha$ 慢慢增加，可以选择改动最小的节点 $\frac{C_\alpha(T_r) - C_\alpha(t_r)}{|T_r|-1}$
- 当选择改动最小那个节点之后，我们可以增大 $\alpha$ ，使得不等式为等式
- 重复迭代这个过程，直到根节点为止，生成一系列备选树

# 决策树效果



# 决策树脑图


决策树
算法
生成算法
ID3，信息增益
C4.5 信息增益率
CART 基尼指数
裁剪
固定权重
迭代生成备选树
优点
构造简单，判别计算快速
对数据不需要任何加工
应对unblance的数据效果好
缺点
泛化能力差，容易出现过拟合
对新增加的样本，需要重新调整树结构
数学知识
熵计算
条件熵

# 集成学习算法

- 集成学习算法(ensemble learning)，两大利器
  - Bagging
    - 随机森林(Random Forest)是Bagging方法的典型
  - Boosting
    - 后续的AdaBoost, GBDT会详细说明

# 随机森林算法

- 通过两个随机性，构造不同的次优树
  - 随机选择样本，通过有放回的采样（Boostrap），重复的选择部分样本来构造树
  - 构造树的过程中，每次随机考察部分特征，不对树进行裁剪
  - 单树采用CART树
- 在生成一定数目的次优树之后，森林的输出采用简单多数投票法（针对分类）或单颗树输出结果的简单平均（针对回归）得到

# 随机森林分析

- 森林中单颗树的分类强度（Strength）：每颗树的分类强度越大，则随机森林的分类性能越好。

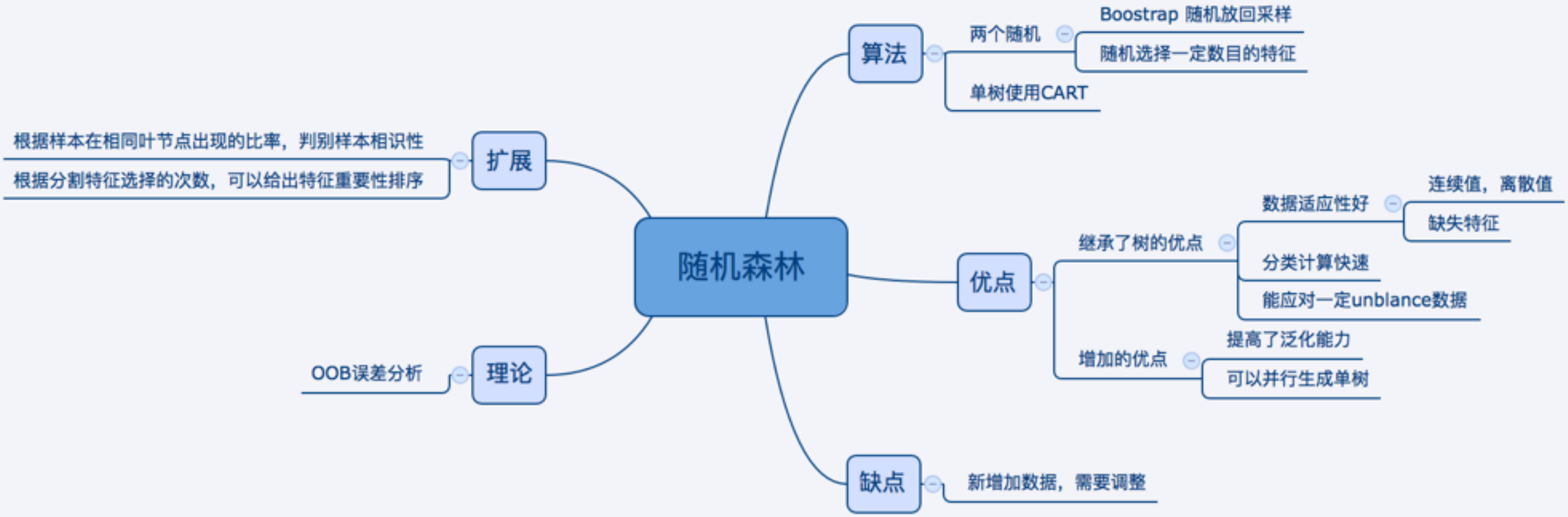
- 森林中树之间的相关度（Correlation）：树之间的相关度越大，则随机森林的分类性能越差。

- OOB错误率是随机森林的错误率无偏估计

  - 对每个样本，在其所有OOB的单树中错误占比，作为OOB错误率

  - 因此随机森林不需要进行交叉验证

# 随机森林的脑图


随机森林
算法
两个随机
Boostrap 随机放回采样
随机选择一定数目的特征
单树使用CART
优点
继承了树的优点
数据适应性好
连续值，离散值
缺失特征
分类计算快速
能应对一定unblance数据
增加的优点
提高了泛化能力
可以并行生成单树
缺点
新增加数据，需要调整
扩展
根据样本在相同叶节点出现的比率，判别样本相识性
根据分割特征选择的次数，可以给出特征重要性排序
理论
OOB误差分析