Decision Tree——CART

课设阶段曾经做过一个decision tree的银行信贷系统，可是直到今天才弄懂decision tree的大概流程并且实现了一下。

一、Decision Tree Hypothesis

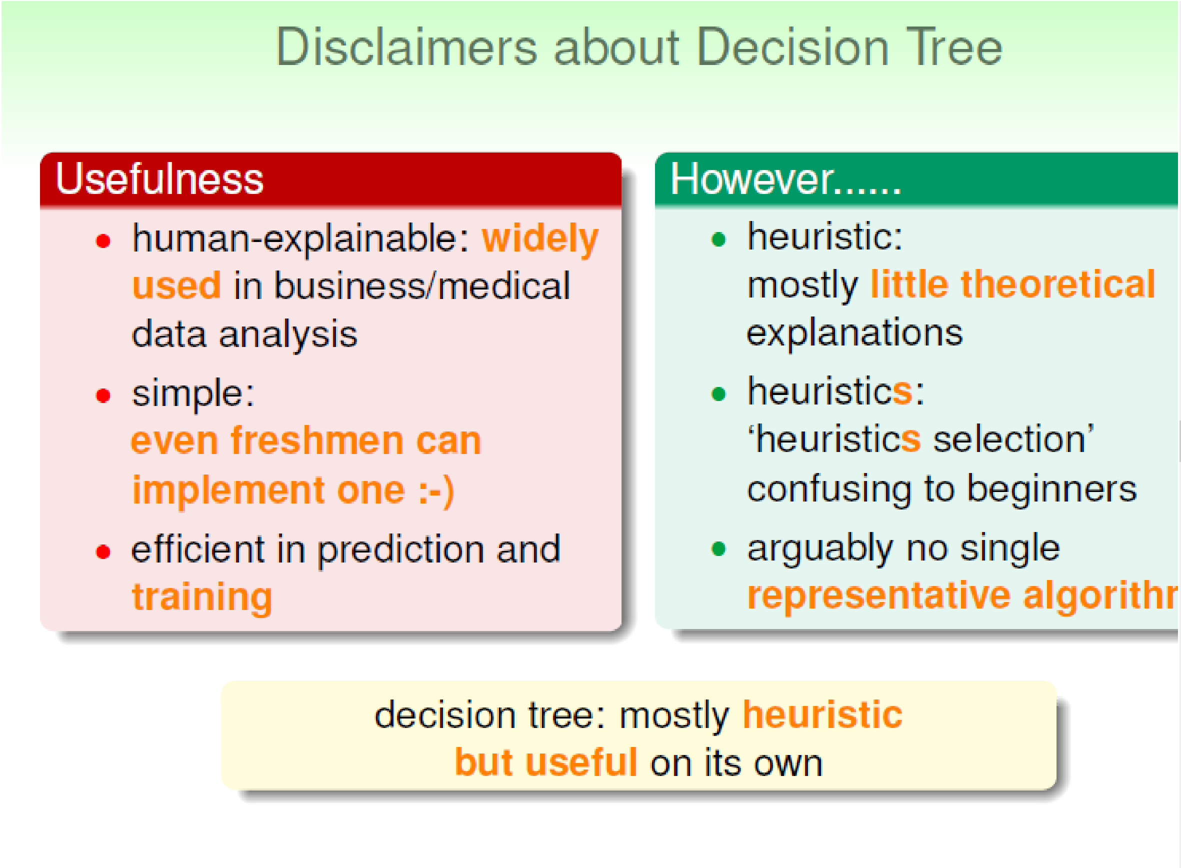
1. 之前提到过关于混合模型aggregation的两种策略，一个是blending，就是对于所有的模型我们在学习好了之后直接拿来用，另一种是learning，就是我们一边学习模型一边采取混合。对于混合的方式，我们有uniform,non-uniform和conditional三种策略。与其对应的总结如下:

   blending: aggregate after getting g_t;
   learning: aggregate as well as getting g_t

aggregation type	blending	learning
uniform	voting/averaging	Bagging
non-uniform	linear	AdaBoost
conditional	stacking	**Decision Tree**

   decision tree: a traditional learning model that realizes **conditional aggregation**

2. 所谓的decision tree就是把一群条件下的东西aggregation起来，类似于人的决策。实现的方法无非就是递归
3. 对于决策树的研究存在着很多启发式的规则，实际上这个是以经验主义为主的模型，很多时候缺乏相应的理论基础。



二、Decision Tree Algorithm

1. 基本的决策树学习流程如下：

$$G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!]\, G_c(\mathbf{x})$$

function DecisionTree(data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$)

if termination criteria met

    return base hypothesis $g_t(\mathbf{x})$

else

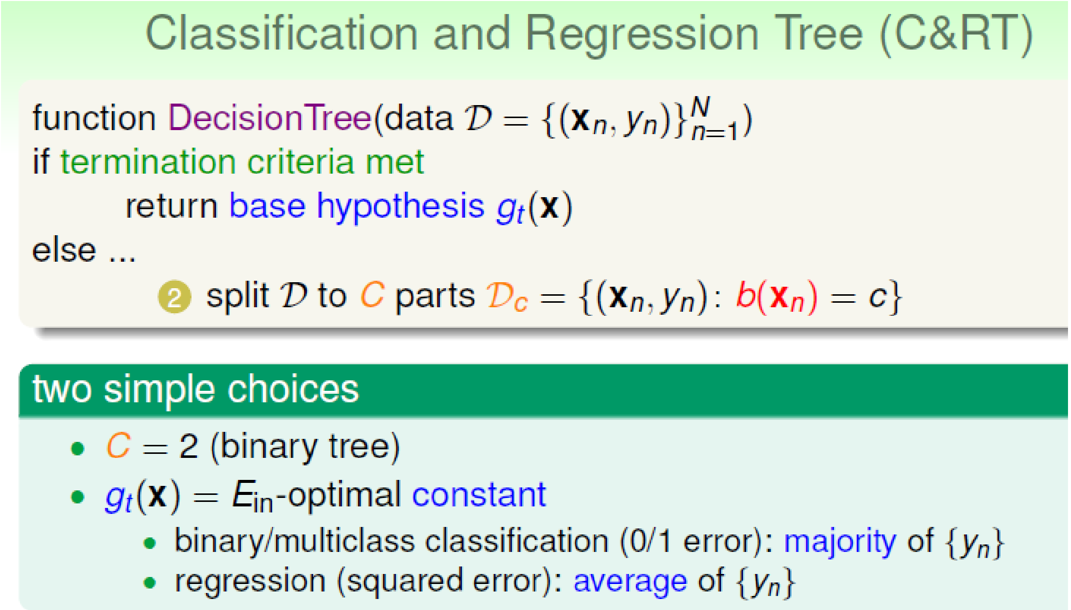
1. learn branching criteria $b(\mathbf{x})$
2. split $\mathcal{D}$ to C parts $\mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\}$
3. build sub-tree $G_c \leftarrow \text{DecisionTree}(\mathcal{D}_c)$
4. return $G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!]\, G_c(\mathbf{x})$

four choices: number of branches, branching criteria, termination criteria, & base hypothesis

对于模型最重要的假设如下：

- 分支数量（叉数）
- 分支的依据（非叶子节点）
- 停止分支的条件
- 基本假设（叶子节点）

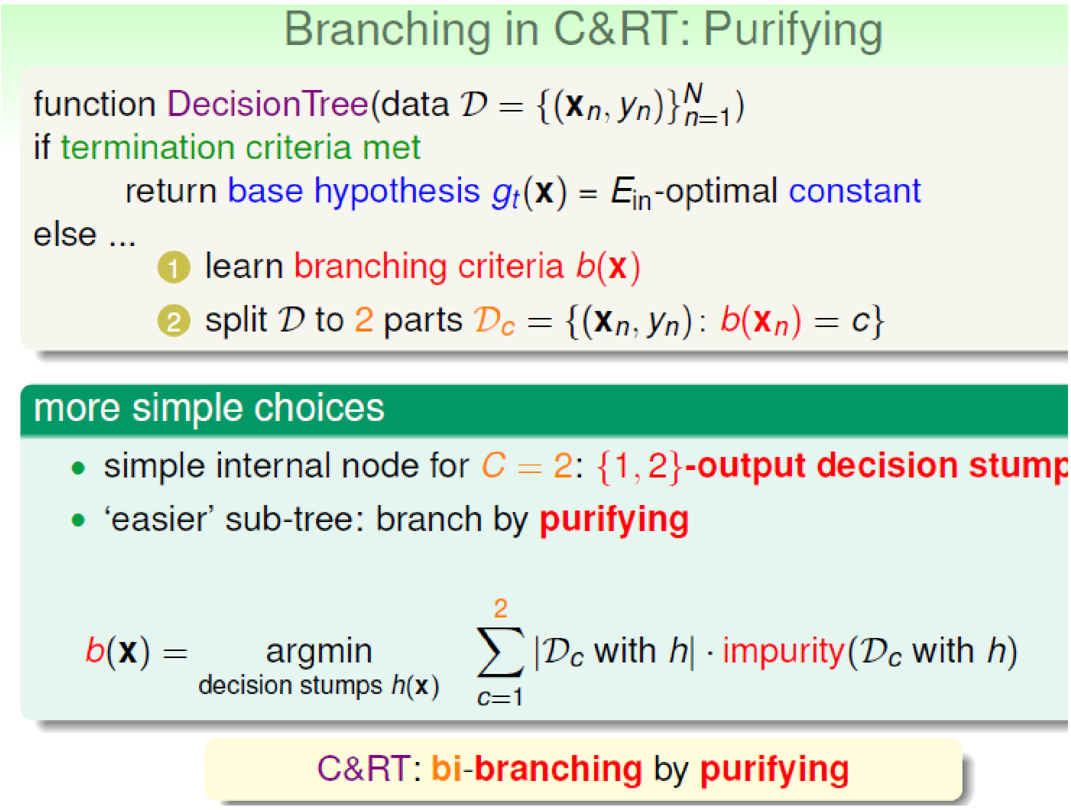
2. CART算法：



首先确定两个指标：

- 分支数量=2
- 基本假设
  - 分类：常数，即出现最多的label
  - 回归：所有label的平均
- 然后是分支依据：

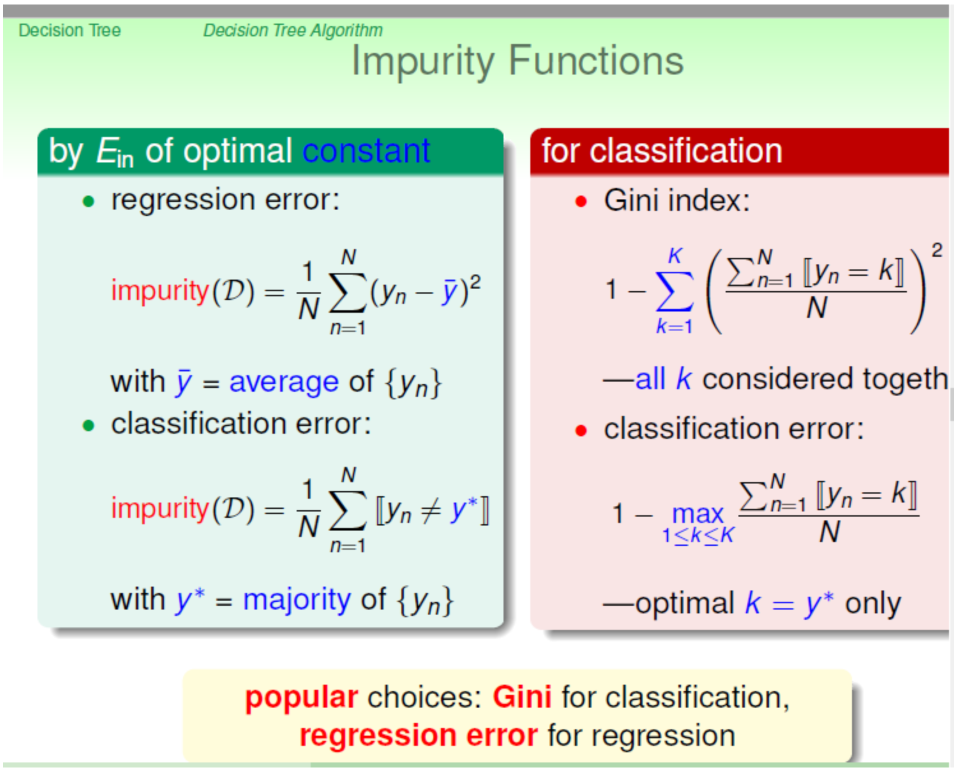
  我们通过优化**不纯度**来进行分支：



不纯度的衡量：

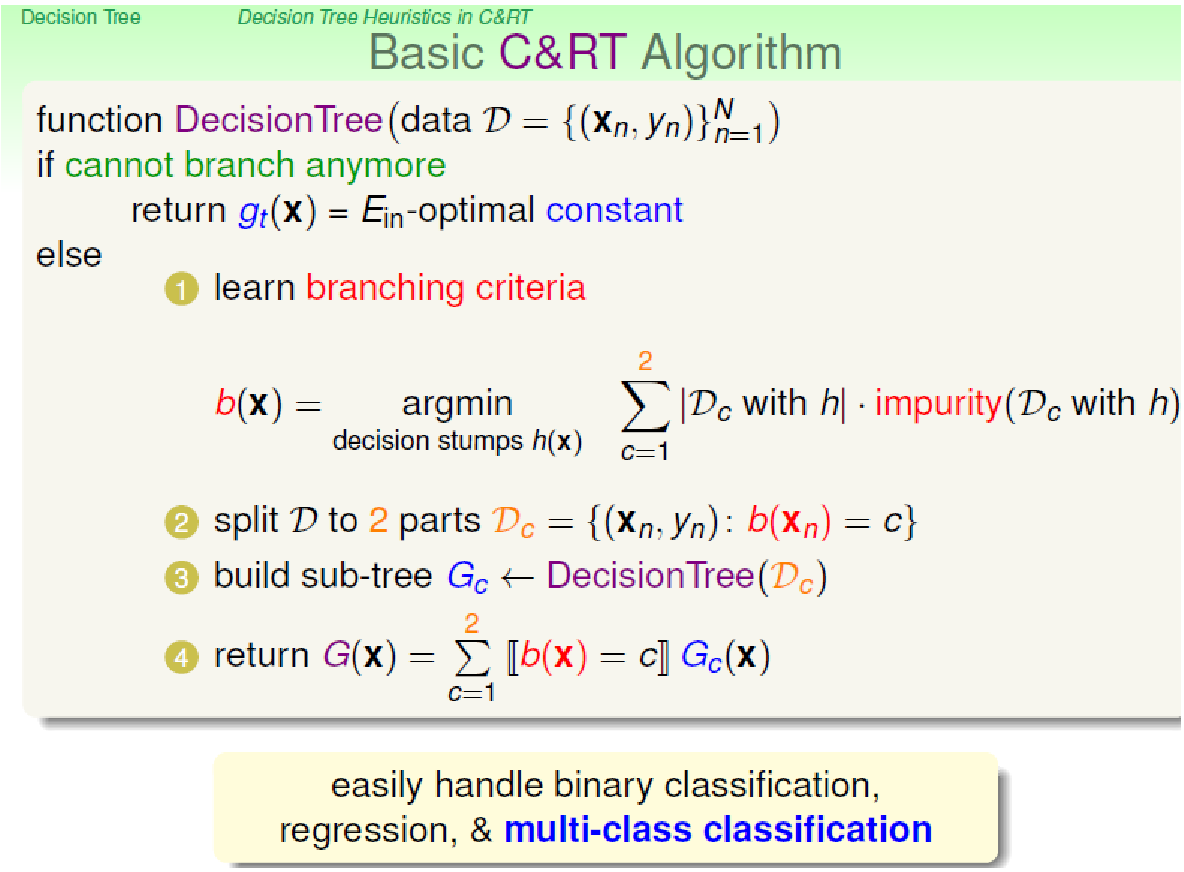
  - 分类问题：基尼指数

- 回归问题：均方误差

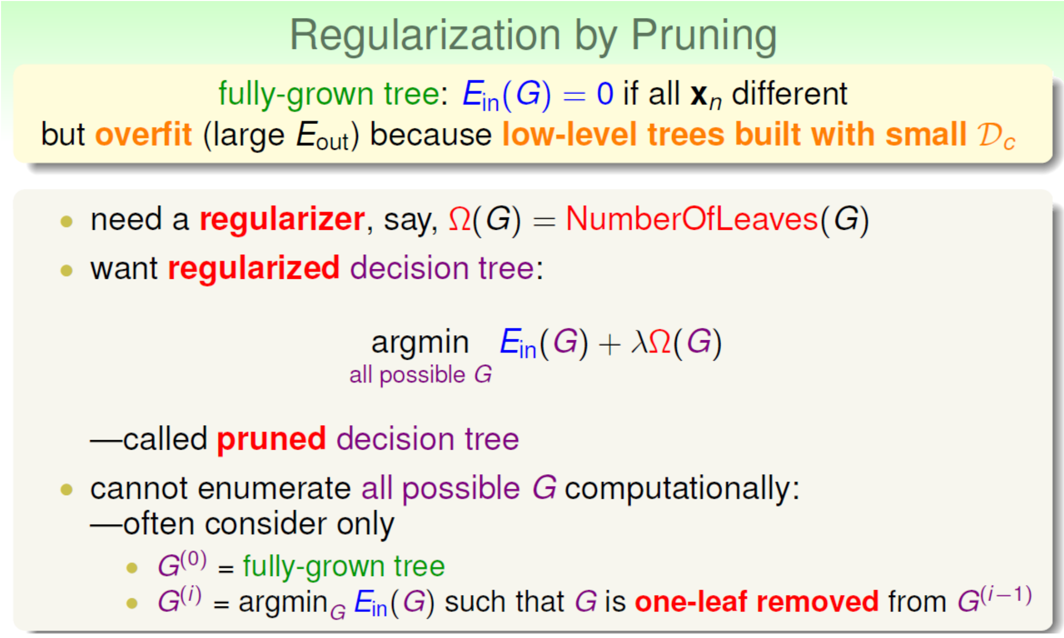


三、Decision Tree Heuristics in CART

1. 基本的CART算法



2. 但是这种树容易过拟合，比如所有的标签都分成一类就ok，这个时候我们需要通过正则化来防止过拟合，思想是希望这棵树的分支越少越好，进而降低模型复杂度



四、Decision Tree in Action

实际上CART回比Adaboost-stump来的更好