

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。

Disclaimers about Decision Tree

Usefulness

- human-explainable: **widely used** in business/medical data analysis
- simple: **even freshmen can implement one :-)**
- efficient in prediction and **training**

However.....

- heuristic: mostly **little theoretical** explanations
- heuristics: 'heuristics selection' confusing to beginners
- arguably no single **representative algorithm**

decision tree: mostly **heuristic**
but useful on its own

二、Decision Tree Algorithm

1. 基本的决策树学习流程如下:

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

```
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  $G(\mathbf{x}) = \sum_{c=1}^C \mathbb{I}[b(\mathbf{x}) = c] G_c(\mathbf{x})$ 
```

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:

Classification and Regression Tree (C&RT)

```
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 ...
    ② split  $\mathcal{D}$  to  $C$  parts  $\mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\}$ 
```

two simple choices

- $C = 2$ (binary tree)
- $g_t(\mathbf{x}) = E_{\text{in}}\text{-optimal constant}$
 - binary/multiclass classification (0/1 error): majority of $\{y_n\}$
 - regression (squared error): average of $\{y_n\}$

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:

Branching in C&RT: Purifying

```
function DecisionTree(data  $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ )
  if termination criteria met
    return base hypothesis  $g_t(\mathbf{x}) = E_{\text{in}}\text{-optimal constant}$ 
  else ...
    ① learn branching criteria  $b(\mathbf{x})$ 
    ② split  $\mathcal{D}$  to 2 parts  $\mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\}$ 
```

more simple choices

- simple internal node for $C = 2$: $\{1, 2\}$ -output decision stump
- 'easier' sub-tree: branch by purifying

$$b(\mathbf{x}) = \underset{\text{decision stumps } h(\mathbf{x})}{\operatorname{argmin}} \sum_{c=1}^2 |\mathcal{D}_c \text{ with } h| \cdot \text{impurity}(\mathcal{D}_c \text{ with } h)$$

C&RT: bi-branching by purifying

不纯度的衡量:

- 分类问题: 基尼指数

- 回归问题：均方误差

Decision Tree
Decision Tree Algorithm

Impurity Functions

by E_{in} of optimal constant

- regression error:

$$\text{impurity}(\mathcal{D}) = \frac{1}{N} \sum_{n=1}^N (y_n - \bar{y})^2$$

with \bar{y} = average of $\{y_n\}$

- classification error:

$$\text{impurity}(\mathcal{D}) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}[y_n \neq y^*]$$

with y^* = majority of $\{y_n\}$

for classification

- Gini index:

$$1 - \sum_{k=1}^K \left(\frac{\sum_{n=1}^N \mathbb{I}[y_n = k]}{N} \right)^2$$

—all k considered together

- classification error:

$$1 - \max_{1 \leq k \leq K} \frac{\sum_{n=1}^N \mathbb{I}[y_n = k]}{N}$$

—optimal $k = y^*$ only

popular choices: **Gini** for classification,
regression error for regression

三、Decision Tree Heuristics in CART

1. 基本的CART算法

Decision Tree
Decision Tree Heuristics in C&RT

Basic C&RT Algorithm

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

if cannot branch anymore

return $g_t(\mathbf{x}) = E_{in}$ -optimal constant

else

① learn branching criteria

$$b(\mathbf{x}) = \underset{\text{decision stumps } h(\mathbf{x})}{\operatorname{argmin}} \sum_{c=1}^2 |\mathcal{D}_c \text{ with } h| \cdot \text{impurity}(\mathcal{D}_c \text{ with } h)$$

② split \mathcal{D} to 2 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 $G(\mathbf{x}) = \sum_{c=1}^2 \mathbb{I}[b(\mathbf{x}) = c] G_c(\mathbf{x})$

easily handle binary classification,
regression, & **multi-class classification**

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

Regularization by Pruning

fully-grown tree: $E_{\text{in}}(G) = 0$ if all \mathbf{x}_n different
but **overfit** (large E_{out}) because **low-level trees built with small \mathcal{D}_c**

- need a **regularizer**, say, $\Omega(G) = \text{NumberOfLeaves}(G)$
- want **regularized** decision tree:

$$\underset{\text{all possible } G}{\operatorname{argmin}} E_{\text{in}}(G) + \lambda \Omega(G)$$

—called **pruned** decision tree

- cannot enumerate **all possible G** computationally:
—often consider only
 - $G^{(0)} = \text{fully-grown tree}$
 - $G^{(i)} = \underset{G}{\operatorname{argmin}} E_{\text{in}}(G)$ such that G is **one-leaf removed** from $G^{(i-1)}$

四、Decision Tree in Action

实际上CART回比Adaboost-stump来的更好