

发件人: Yulin Wu yw4923@nyu.edu
 主题:
 日期: 2020年9月9日 下午7:04
 收件人:



Decision Tree

2020年9月8日 星期二
 下午3:32

What We Have Done		
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

conditional: $\sum_{t=1}^T \alpha_t(x) g_t(x)$, α_t 和 x 有关系, 那么使用非线性 g_t 的组合就行

learning: 原本不知道有哪些小 g , 我们要学习这些小 g

bagging: 用 bootstrap 将我们的数据变成不一样的副本, 最后再 uniform 地合起来

adaboost: 放大错误集的权重, 且同时 learn α (不同的票数)

Decision Tree : 决策树

Decision Tree for Watching MOOC Lectures

$G(\mathbf{x}) = \sum_{t=1}^T q_t(\mathbf{x}) \cdot g_t(\mathbf{x})$

- base hypothesis $g_t(\mathbf{x})$: leaf at end of path t , a **constant** here
- condition $q_t(\mathbf{x})$: [is \mathbf{x} on path t ?]
- usually with **simple internal nodes**

decision tree: arguably one of the most **human-mimicking models**

决策树是一个集成模型。

$g_t(\mathbf{x})$ 就是我们之前常说的基础模型, $g_t(\mathbf{x}) = 0$ (No) or 1 (yes).

$q_t(\mathbf{x})$ 是权重, conditional: 给定了 $\mathbf{x} = [\text{quitting time, has a date, deadline}]$ 的情况。

将树拆分 :

Recursive View of Decision Tree

Path View: $G(\mathbf{x}) = \sum_{t=1}^T [\mathbf{x} \text{ on path } t] \cdot \text{leaf}_t(\mathbf{x})$

Recursive View

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

- $G(\mathbf{x})$: full-tree hypothesis
- $b(\mathbf{x})$: branching criteria
- $G_c(\mathbf{x})$: sub-tree hypothesis at the c -th branch

A Basic Decision Tree Algorithm

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

```

function DecisionTree(data  $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ )
  if termination criteria met
    return base hypothesis  $g(\mathbf{x})$ 
  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

cart : bi-branching by purifying

two simple choices

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

好滴, 现在还要决定 termination criteria and branching criteria

branching criteria:

more simple choices

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

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

|D_c with h|是资料的权重，如：资料的大小比较大的话，这个资料更重要

termination criteria:

'forced' to terminate when

- all y_n the same: $\text{impurity} = 0 \Rightarrow g_t(\mathbf{x}) = y_n$
- all x_n the same: **no decision stumps**

regularization (剪枝)

Regularization by Pruning

fully-grown tree: $E_{in}(G) = 0$ if all x_n different
but **overfit** (large E_{out}) because **low-level trees built with small D_C**

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

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

—called **pruned** decision tree

- cannot enumerate **all possible** G computationally:

—often consider only

- $G^{(0)}$ = fully-grown tree
- $G^{(i)} = \underset{G}{\operatorname{argmin}} E_{in}(G)$ such that G is **one-leaf removed** from $G^{(i-1)}$

$G^{(1)}$ $G^{(2)}$

当数据缺失的时候，可以使用替代的feature

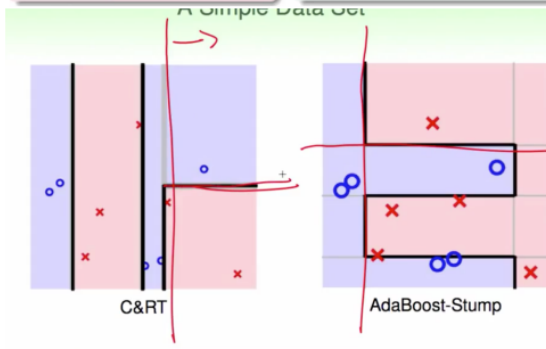
优缺点

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**



已使用 OneNote 创建。

by E_{in} of optimal constant	for classification
<ul style="list-style-type: none"> regression error: $\text{impurity}(D) = \frac{1}{N} \sum_{n=1}^N (y_n - \bar{y})^2$ <p>with \bar{y} = average of $\{y_n\}$</p> <ul style="list-style-type: none"> classification error: $\text{impurity}(D) = \frac{1}{N} \sum_{n=1}^N [y_n \neq y^*]$ <p>with y^* = majority of $\{y_n\}$</p>	<ul style="list-style-type: none"> Gini index: $1 - \sum_{k=1}^K \left(\frac{\sum_{n=1}^N [y_n = k]}{N} \right)^2$ <p>—all k considered together</p> <ul style="list-style-type: none"> classification error: $1 - \max_{1 \leq k \leq K} \frac{\sum_{n=1}^N [y_n = k]}{N}$ <p>—optimal $k = y^*$ only</p>
<p>popular choices: Gini for classification, regression error for regression</p>	

but enumerate all possible G is computationally complicated
so 使用后置剪枝：先fit出最好的 G ，再摘掉一片叶子，再摘掉第二片叶子。。。