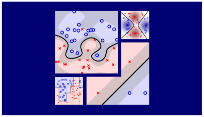
Machine Learning Techniques
(機器學習技法)



Lecture 9: Decision Tree

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 8: Adaptive Boosting

optimal re-weighting for diverse hypotheses and adaptive **linear aggregation** to **boost 'weak' algorithms**

Lecture 9: Decision Tree

- Decision Tree Hypothesis
- Decision Tree Algorithm
- Decision Tree Heuristics in C&RT
- Decision Tree in Action

3. Distilling Implicit Features: Extraction Models

What We Have Done

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

已经有gt

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

通过bootsttap抽样得到不同样本Dt. 学习等权重的不同gt

不同批次的样本权重不同，每次权重按上次结果调整。学习不同gt及对应权重at。按权重线性组合

用线性/非线性模型学习权重at，组合已有gt

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

不同条件下， 不同的权重。gt权重与条件有关

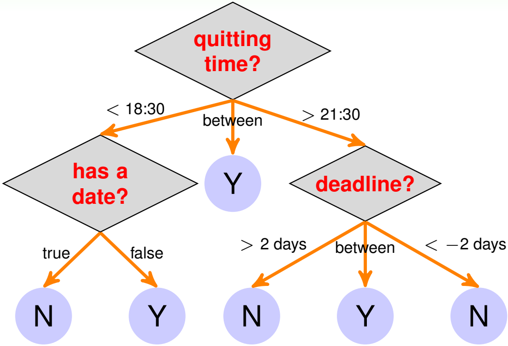
Decision Tree for Watching MOOC Lectures

看作是 g1(x)=N q1(x)=(<18:30 && true)
g2(x)=Y q2(x)=(<18:30 && false)
g3(x)=Y q3(x)=(18:30<time<21:30)
... g6(x)=N q6(x)=...

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

- **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**

相当于|leaf|个有条件的函数，组合起来

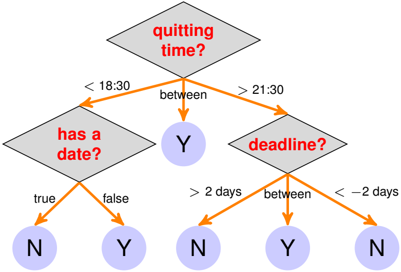


要不要看线上课程。二分类。3个特征去判断

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

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 递归

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

各个分支的条件　子树

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

tree = (root, sub-trees), just like what **your data structure instructor would say :-)**

Disclaimers about Decision Tree

可解释性

缺点：理论保证少一点

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
- heuristic**s**: ‘heuristic**s** selection’ confusing to beginners
- arguably no single **representative algorithm**

没有具有代表性的，都是历史上流行或者不流行

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

主要是有用

Fun Time

The following C-like code can be viewed as a decision tree of three leaves.

```
if (income > 100000) return true;
else {
  if (debt > 50000) return false;
  else return true;
}
```

What is the output of the tree for (`income`, `debt`) = (98765, 56789)?

1. true
2. false
3. 98765
4. 56789

Fun Time

The following C-like code can be viewed as a decision tree of three leaves.

```
if (income > 100000) return true;
else {
  if (debt > 50000) return false;
  else return true;
}
```

What is the output of the tree for ($\mathtt{income}$, $\mathtt{debt}$) = (98765, 56789)?

1. true
2. false
3. 98765
4. 56789

Reference Answer: 2

You can simply trace the code. The tree expresses a complicated boolean condition $[\![\mathtt{income} > 100000 \text{ or } \mathtt{debt} \leq 50000]\!]$.

A Basic Decision Tree Algorithm

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

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

if termination criteria met

  return base hypothesis $g_t(\mathbf{x})$ 到停止条件

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 $G_c \leftarrow$ DecisionTree($\mathcal{D}_c$)
4. return $G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!]\, G_c(\mathbf{x})$

分几枝 如何分支最好拟合train等。

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

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) 二叉树
- $g_t(\mathbf{x}) = E_{\text{in}}$-optimal constant train落到这里以后，占多数的类别
  - binary/multiclass classification (0/1 error): majority of $\{y_n\}$ 回传 类别
  - regression (squared error): average of $\{y_n\}$ 实值

或者落到这里数据的平均

disclaimer:
C&RT here is based on **selected components** of **CART^{TM} of California Statistical Software**

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_{in}-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**

用decision stump 切分，把D切分为两类
yc=s* (xi-theta) （按某一维某个theta）
theta是所有特征的所有可能切分点

如何选择特征和对应的切分点呢？
希望切完后，纯度较高。不纯度降低。选纯度最高的切法

选切完后，不纯度最低的h(x)

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

考虑decision stump所有可能的切分点

D1如果比较大
D1纯不纯比较重要
用|D1|/|D|加权

D1 ,D2各自的不纯度。不纯度越低越好

C&RT: bi-branching by purifying 纯化

Impurity Functions

所有类别k所占比例之和,越大越纯
如果只有一种数据，1
全是不同数据，1/k

Gini(D)=1-sum_i (pi**2)

by E_{in} of optimal constant

- regression error:

方差越大越不纯

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

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

- classification error:

受D的众数影响大

与多数不一样的越多，越不纯

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

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

选切分后，加权不纯度最小的切分方法回传

for classification

- Gini index: Gini 指数：不纯度，越大越不纯
切分后的加权不纯度越低越好

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

考虑所有类别的纯度

—all k considered together

- classification error:

$$1 - \max_{1\le k\le K}\frac{\sum_{n=1}^{N}[\![y_n = k]\!]}{N}$$

—optimal $k = y^*$ only

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

Termination in C&RT

function DecisionTree(data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$)
if termination criteria met
  return base hypothesis $g_t(\mathbf{x})$ = E_{in}-optimal constant
else ...
  1. learn branching criteria

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

'forced' to terminate when

- all y_n the same: impurity = 0 $\Longrightarrow g_t(\mathbf{x}) = y_n$ 如果最后的加权不纯度很低了 很纯 b(x)==0 停止 一般b(x)< e
- all $\mathbf{x}_n$ the same: **no decision stumps** 或者落到这一点的x都一样了(rare,或者只有一个x) 没法做decision stump了

C&RT: **fully-grown tree** with constant leaves that come from **bi-branching** by **purifying** 完全树

Fun Time

For the Gini index, $1 - \sum_{k=1}^{K} \left(\frac{\sum_{n=1}^{N} [\![y_n = k]\!]}{N} \right)^2$. Consider $K = 2$, and let $\mu = \frac{N_1}{N}$, where N_1 is the number of examples with $y_n = 1$. Which of the following formula of μ equals the Gini index in this case?

1. $2\mu(1-\mu)$
2. $2\mu^2(1-\mu)$
3. $2\mu(1-\mu)^2$
4. $2\mu^2(1-\mu)^2$

Fun Time

For the Gini index, $1 - \sum_{k=1}^{K} \left(\frac{\sum_{n=1}^{N} [\![y_n = k]\!]}{N} \right)^2$. Consider $K = 2$, and let $\mu = \frac{N_1}{N}$, where N_1 is the number of examples with $y_n = 1$. Which of the following formula of μ equals the Gini index in this case?

1. $2\mu(1 - \mu)$
2. $2\mu^2(1 - \mu)$
3. $2\mu(1 - \mu)^2$
4. $2\mu^2(1 - \mu)^2$

Reference Answer: 1

Simplify $1 - (\mu^2 + (1 - \mu)^2)$ and the answer should pop up.

Basic C&RT Algorithm

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

if cannot branch anymore

return $g_t(\mathbf{x})$ = E_{in}-optimal constant train好的叶子类别/train好的均值

else

1. learn branching criteria

desicion stump 选加权不纯度最低的切分点

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

二分

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

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

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:

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

选拟合不错，叶子节点不多的树

—called **pruned** decision tree

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

G(0)有10片叶子。G(1):剪一个内部节点 G(2):剪2个内部节点
G(T):只剩下一个节点
根据vaidation, 从不同大小的树中，选择一颗最好的G(i)

如何删节点？ 剪掉一个内部节点t, t的子树Tt全部合并,成为一个叶子节点t。 (之前Tt的数据全部落在t)
选择误差增加率最小的节点进行剪枝: R(t):剪枝后t节点的预测错误样本。R(Tt):剪枝前子树Tt的所有错误样本
误差增加率： R|t|- R|Tt| / (|Tt|-1) 错误增加/叶子节点减少数目
CCP算序列Gi时误差用train
选序列Gi用valid交叉验证

systematic choice of λ? **validation**

Branching on Categorical Features

Xi是非数字特征。decision stump不好切割

numerical features

blood pressure:
130, 98, 115, 147, 120

categorical features

major symptom:
fever, pain, tired, sweaty

branching for numerical

decision stump

$$b(\mathbf{x}) = [\![x_i \le \theta]\!] + 1$$

with $\theta \in \mathbb{R}$

branching for categorical

decision subset

$$b(\mathbf{x}) = [\![x_i \in S]\!] + 1$$

with $S \subset \{1, 2, \ldots, K\}$

特征Xi有K种, 可以穷举所有的自集。S={1}/.../{K}/{1,2}/{1,3}/...
按原始数据Xn的该维Xn,d是否在S中，将X分割为两类

C&RT (& general decision trees):
handles **categorical features easily**

类似于
decision
stump

Missing Features by Surrogate Branch

处理预测时的缺失数据：用关联性强的surrogater代替原始的分割器b(x)

possible $b(\mathbf{x}) = [\![\text{weight} \le 50\text{kg}]\!]$

if weight missing during prediction: 如果test

- what would human do?
  - go get weight
  - or, use threshold on height instead, because threshold on height ≈ threshold on weight
- surrogate branch:
  - maintain surrogate branch $b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots \approx$ best branch $b(\mathbf{x})$ during training
  - allow missing feature for $b(\mathbf{x})$ during prediction by using surrogate instead

当一个变量作为competitor时，不关心primary node的分割结果。可以用多个变量代替当前的单一变量进行分割

但作为surrogater,就需要模仿了

改切别的特征

训练时，用多个代理属性，达到和原始属性b(x)相近的效果

测试时，当某个样本特征b(x)缺失时，用代替特征进行切割

如果5-10个surrogater中,某个差于default rule,舍弃。相关性score<0

训练时，每个分割点node都会找5个surrogater split以及对应的分割点。当primary缺失时，按照相关性，用surrogater 1 分割。如果这个也缺失了，用surogater2分割。从而防止预测时有缺失。

C&RT: handles missing features easily

如果surrogate与primary spliter负相关很强。用反向的结果

surrogater只用一个变量（尽管primary spliter可以是多变量的组合）。可以是连续变量或者类别，且只分两类

尽可能模拟primary spliter的分割行为，而非这个变量本身。

相关性:和defult rule相比，错误率的相对提升 如果training数据在surrogate和主spliter的分割结果完全相同,r==1

defualt rule: 如果主spliter 60 40 ，那么让数据全分到majority class。错误率40% surrogater: 90,10 err:30% 提升0.25

surrogater 60,40 error:0 提升 1

Fun Time

For a categorical branching criteria $b(\mathbf{x}) = [\![x_i \in S]\!] + 1$ with $S = \{1, 6\}$. Which of the following is the explanation of the criteria?

1. if *i*-th feature is of type 1 or type 6, branch to first sub-tree; else branch to second sub-tree
2. if *i*-th feature is of type 1 or type 6, branch to second sub-tree; else branch to first sub-tree
3. if *i*-th feature is of type 1 and type 6, branch to second sub-tree; else branch to first sub-tree
4. if *i*-th feature is of type 1 and type 6, branch to first sub-tree; else branch to second sub-tree

Fun Time

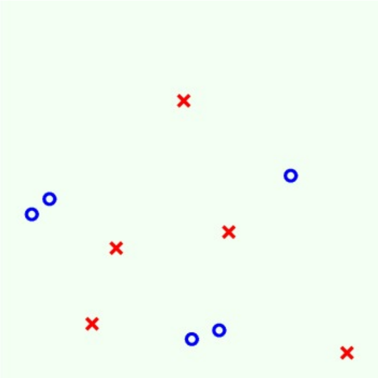
For a categorical branching criteria $b(\mathbf{x}) = [\![x_i \in S]\!] + 1$ with $S = \{1, 6\}$. Which of the following is the explanation of the criteria?

1. if i-th feature is of type 1 or type 6, branch to first sub-tree; else branch to second sub-tree
2. if i-th feature is of type 1 or type 6, branch to second sub-tree; else branch to first sub-tree
3. if i-th feature is of type 1 and type 6, branch to second sub-tree; else branch to first sub-tree
4. if i-th feature is of type 1 and type 6, branch to first sub-tree; else branch to second sub-tree

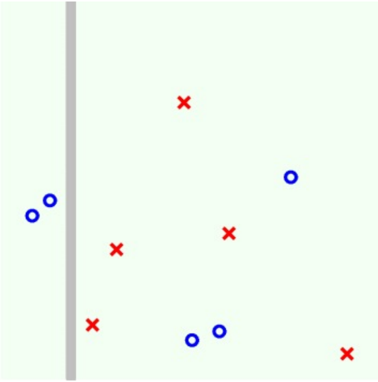
Reference Answer: 2

Note that '$\in S$' is an 'or'-style condition on the elements of S in human language.

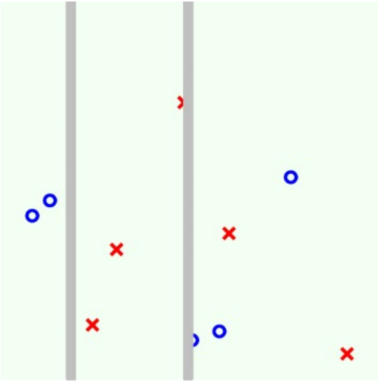
A Simple Data Set



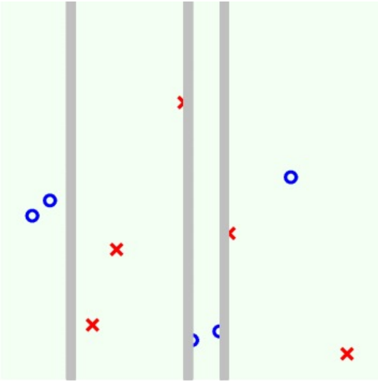
A Simple Data Set



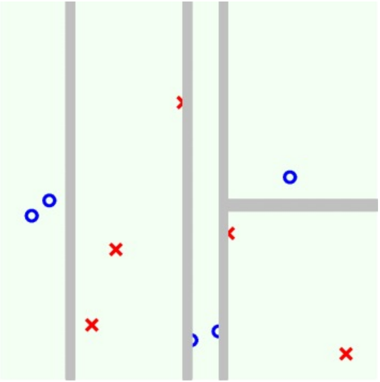
A Simple Data Set



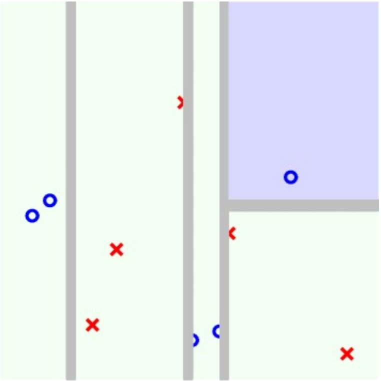
A Simple Data Set



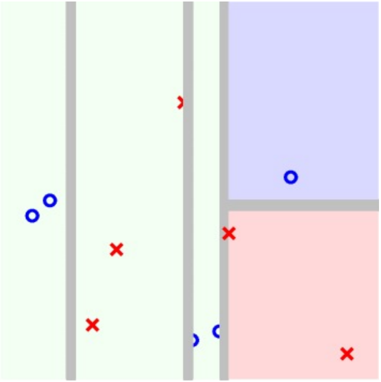
A Simple Data Set



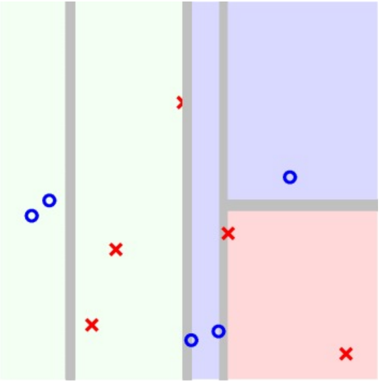
A Simple Data Set



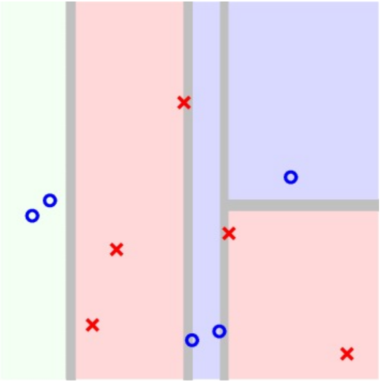
A Simple Data Set



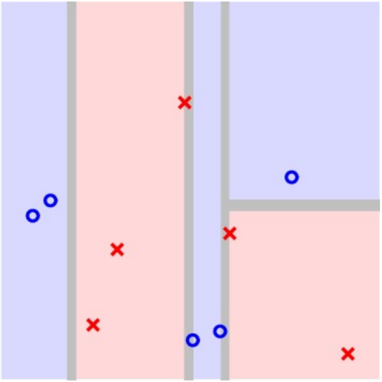
A Simple Data Set



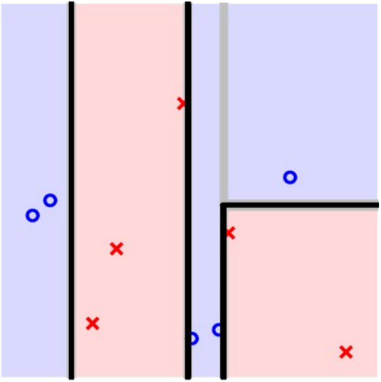
A Simple Data Set



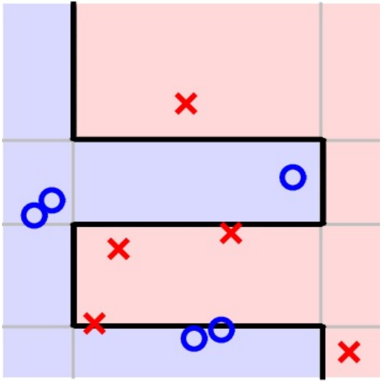
A Simple Data Set



A Simple Data Set



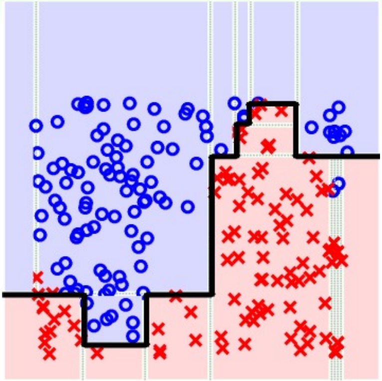
C&RT



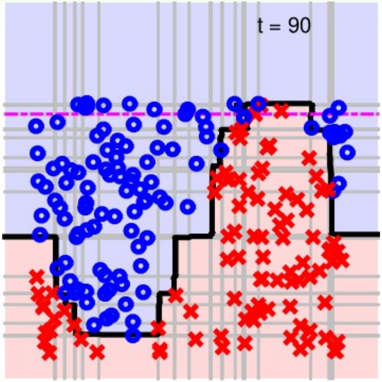
AdaBoost-Stump

C&RT: 'divide-and-conquer'

A Complicated Data Set



C&RT



AdaBoost-Stump

比Adaboost更好

C&RT: even more efficient than AdaBoost-Stump

Practical Specialties of C&RT

- human-explainable
- multiclass easily
- categorical features easily
- missing features easily
- efficient non-linear training (and testing)

—almost no other learning model share all such specialties, except for **other decision trees**

another popular decision tree algorithm:
C4.5, with different **choices of heuristics**

Fun Time

Which of the following is **not** a specialty of C&RT without pruning?

1. handles missing features easily
2. produces explainable hypotheses
3. achieves low E_{in}
4. achieves low E_{out}

Fun Time

Which of the following is **not** a specialty of C&RT without pruning?

1. handles missing features easily
2. produces explainable hypotheses
3. achieves low E_{in}
4. achieves low E_{out}

Reference Answer: 4

The first two choices are easy; the third comes from the fact that fully grown C&RT greedy minimizes E_{in} (almost always to 0). But as you may imagine, overfitting may happen and E_{out} may not always be low.

Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 9: Decision Tree

- Decision Tree Hypothesis
  express path-conditional aggregation
- Decision Tree Algorithm
  recursive branching until termination to base
- Decision Tree Heuristics in C&RT
  pruning, categorical branching, surrogate
- Decision Tree in Action
  explainable and efficient

- **next: aggregation of aggregation?!**

3. Distilling Implicit Features: Extraction Models