Decision Trees

Fabrice Rossi

SAMM
Université Paris 1 Panthéon Sorbonne

2018

Outline

Trees and decision trees

Growing a tree

Pruning a tree

Machine learning

Supervised learning

- given a data set $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$
- an algorithm builds a program that maps inputs to desired outputs

A simple example

	Color	Size	Action	Age	Class
1	YELLOW	SMALL	STRETCH	ADULT	TRUE
2	YELLOW	SMALL	STRETCH	CHILD	TRUE
3	YELLOW	SMALL	DIP	ADULT	TRUE
4	YELLOW	SMALL	DIP	CHILD	TRUE
5	YELLOW	LARGE	STRETCH	ADULT	TRUE
6	YELLOW	LARGE	STRETCH	CHILD	FALSE
7	YELLOW	LARGE	DIP	ADULT	FALSE
8	YELLOW	LARGE	DIP	CHILD	FALSE
9	PURPLE	SMALL	STRETCH	ADULT	TRUE
10	PURPLE	SMALL	STRETCH	CHILD	FALSE
11	PURPLE	SMALL	DIP	ADULT	FALSE
12	PURPLE	SMALL	DIP	CHILD	FALSE
13	PURPLE	LARGE	STRETCH	ADULT	TRUE
14	PURPLE	LARGE	STRETCH	CHILD	FALSE
15	PURPLE	LARGE	DIP	ADULT	FALSE
16	PURPLE	LARGE	DIP	CHILD	FALSE

Perfect solution

```
function CLASS(Color, Size, Action, Age)
    if Color=Yellow and Size=Small then
        return True
    else
        if Action=Stretch and Age=Adult then
            return True
        else
            return False
        end if
    end if
end function
```

Simplified programs

In this example

- a series of tests (questions)
- each test consists in matching the value of a variable with a target value
- could be rewritten to get rid of the `and` operator
- in many cases no loops are required

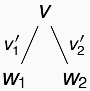
Building programs

- use this simple representation for machine learning
- build a model as a series of tests followed by a “decision”

Tree based representation

Trees

- numerous (equivalent) mathematical definitions
- simple recursive one: a tree consists in a value v and a possibly empty list of trees, each named by a value v' (they are *sub-trees*)
- examples:
  - $(v, \emptyset)$, simplified into v
  - $(v, ((v'_1, (w_1, \emptyset)), (v'_2, (w_2, \emptyset))))$, simplified into $(v, v'_1 \Rightarrow w_1, v'_2 \Rightarrow w_2)$ or better into

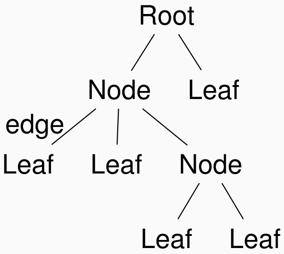


- notice that labelling the sub-trees is application specific (many do not use such labels)

Vocabulary

- each tree consists of a root node and of possibly sub-trees
- a tree without sub-trees is a leaf
- the root node of a sub-tree in a larger tree is simple called a node
- an edge materializes the connection between a tree and a sub-tree and is labelled by the name of the sub-tree

a tree of height 3



- a branch is a series of edges going down the tree
- the height of a tree is the number of edges of its longest branch

Decision trees

Informal definition

A decision tree is a simplified program represented by a tree in which a series of tests are conducted against an input. Each test is represented by a node. Edges correspond to test results and leaves to outputs (a.k.a. decisions).

Decision trees

Informal definition

A decision tree is a simplified program represented by a tree in which a series of tests are conducted against an input. Each test is represented by a node. Edges correspond to test results and leaves to outputs (a.k.a. decisions).

	Color	Size	Action	Age	Class
1	YELLOW	SMALL	STRETCH	ADULT	TRUE
2	YELLOW	SMALL	STRETCH	CHILD	TRUE
3	YELLOW	SMALL	DIP	ADULT	TRUE
4	YELLOW	SMALL	DIP	CHILD	TRUE
5	YELLOW	LARGE	STRETCH	ADULT	TRUE
6	YELLOW	LARGE	STRETCH	CHILD	FALSE
7	YELLOW	LARGE	DIP	ADULT	FALSE
8	YELLOW	LARGE	DIP	CHILD	FALSE
9	PURPLE	SMALL	STRETCH	ADULT	TRUE
10	PURPLE	SMALL	STRETCH	CHILD	FALSE
11	PURPLE	SMALL	DIP	ADULT	FALSE
12	PURPLE	SMALL	DIP	CHILD	FALSE
13	PURPLE	LARGE	STRETCH	ADULT	TRUE
14	PURPLE	LARGE	STRETCH	CHILD	FALSE
15	PURPLE	LARGE	DIP	ADULT	FALSE
16	PURPLE	LARGE	DIP	CHILD	FALSE

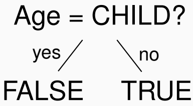
```
if Age=Child then
    return False
else
    return True
end if
```

Decision trees

Informal definition

A decision tree is a simplified program represented by a tree in which a series of tests are conducted against an input. Each test is represented by a node. Edges correspond to test results and leaves to outputs (a.k.a. decisions).

	Color	Size	Action	Age	Class
1	YELLOW	SMALL	STRETCH	ADULT	TRUE
2	YELLOW	SMALL	STRETCH	CHILD	TRUE
3	YELLOW	SMALL	DIP	ADULT	TRUE
4	YELLOW	SMALL	DIP	CHILD	TRUE
5	YELLOW	LARGE	STRETCH	ADULT	TRUE
6	YELLOW	LARGE	STRETCH	CHILD	FALSE
7	YELLOW	LARGE	DIP	ADULT	FALSE
8	YELLOW	LARGE	DIP	CHILD	FALSE
9	PURPLE	SMALL	STRETCH	ADULT	TRUE
10	PURPLE	SMALL	STRETCH	CHILD	FALSE
11	PURPLE	SMALL	DIP	ADULT	FALSE
12	PURPLE	SMALL	DIP	CHILD	FALSE
13	PURPLE	LARGE	STRETCH	ADULT	TRUE
14	PURPLE	LARGE	STRETCH	CHILD	FALSE
15	PURPLE	LARGE	DIP	ADULT	FALSE
16	PURPLE	LARGE	DIP	CHILD	FALSE



Age = CHILD?

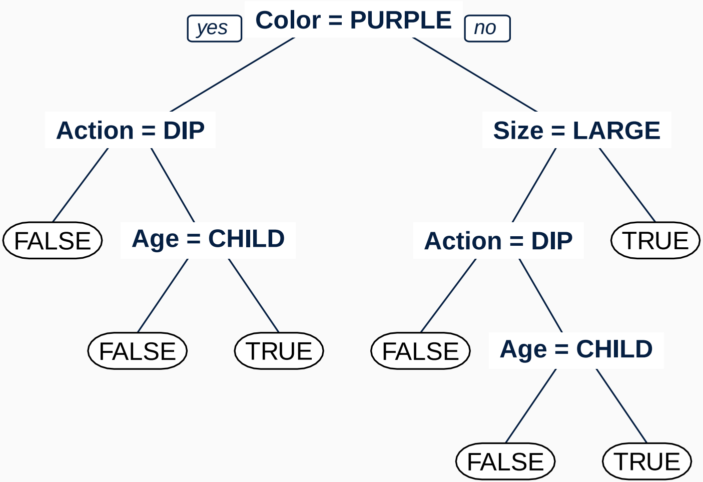
yes

no

FALSE

TRUE

A more complex example



Decision trees

Standard data spaces

- $\mathcal{X}$: input space ($\mathbf{X} \in \mathcal{X}$)
- $\mathcal{Y}$: output space ($\mathbf{Y} \in \mathcal{Y}$)

Definition

A decision tree on $\mathcal{X} \times \mathcal{Y}$ is a tree such that:

- each node n is associated to a function f_n from $\mathcal{X}$ to $\mathcal{O}_n$ a finite set
- a node n has exactly $|\mathcal{O}_n|$ sub-trees with labelled with distinct elements of $\mathcal{O}_n$
- each leaf's value is an element of $\mathcal{Y}$

$\mathcal{T}(\mathcal{X}, \mathcal{Y})$: the set of all decision trees on $\mathcal{X} \times \mathcal{Y}$.

Taking decisions

Using a decision tree

- a decision tree T implements a function g_T from $\mathcal{X}$ to $\mathcal{Y}$
- given $\mathbf{x}$, one navigates from the top of the tree to a leaf, according to test results
- $g_T(\mathbf{x})$ is the value associated to the leaf

Formal definition

$g_T(\mathbf{x})$ is given by:

- $\mathbf{y}$ if $T = \mathbf{y}$ (a tree which consists in a single leaf with value $\mathbf{y}$)
- $g_{T_l}(\mathbf{x})$ if $T = (f_n, (o_1, T_1), \ldots, (o_k, T_k))$ where $o_l = f_n(\mathbf{x})$. Indeed in this case
  - T has for value the function f_n and has as many sub-trees as possible outputs of f_n
  - then T_l is the sub-tree associated $f_n(\mathbf{x})$

Data model

Multivariate assumption

$\mathcal{X}$ is structured into “variables”:

- $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_P$
- $\mathbf{X} = (X_1, X_2, \ldots, X_P)^T$
- $\mathbf{x} = (x_1, x_2, \ldots, x_P)^T$
- the test in a decision tree node operates only on one variable

Variable types

- $\mathcal{X}_k$ is either $\mathbb{R}$ or finite
- different variables can have different types (mixed data)
- $\mathcal{Y}$ is also either $\mathbb{R}$ or finite

Tests

Trivial tests

- $f_n(\mathbf{x}) = x_j$
- only when $|\mathcal{X}_j| < \infty$
- ID3/C4.5 algorithms (Quinlan)

Binary tests

- 1/0, yes/no, true/false output
- $f_n(\mathbf{x}) = \mathbf{1}_{x_j \in A}$ with $A \subset \mathcal{X}_j$ when $|\mathcal{X}_j| < \infty$
- $f_n(\mathbf{x}) = \mathbf{1}_{x_j \leq \lambda}$ when $\mathcal{X}_j = \mathbb{R}$
- CART (Breiman et al.)

Building a decision tree

Supervised learning

- given a data set $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$ and a loss function l
- how to build a tree T such that $R_l(g_T)$ is as small as possible?

Key idea

Recursively grow the tree

- split the data set into sub data-sets based on a test
- recursively apply a tree growing algorithm on each sub-set
- join the sub-trees into a tree with the test at its root

Outline

Trees and decision trees

Growing a tree

Pruning a tree

Optimal leaf

Single leaf tree

- if $T = \mathbf{v}$, then $g_T(\mathbf{x}) = \mathbf{v}$ for all $\mathbf{x}$
- then
$$\widehat{R}_l(g_T, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} l(\mathbf{v}, \mathbf{y})$$
- easy to solve for $\widehat{\mathbf{y}}^*_l(\mathcal{D})$ that minimizes $\widehat{R}_l(g_T, \mathcal{D})$

Examples

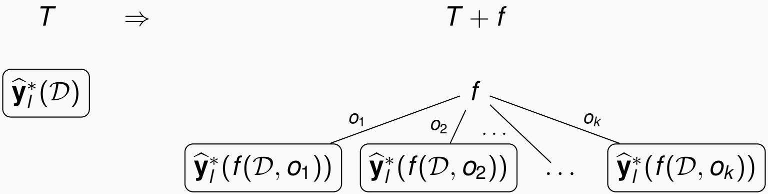
- $\widehat{\mathbf{y}}^*_{l_2}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbf{y}$ when $l_2(p, v) = (p - v)^2$
- $\widehat{\mathbf{y}}^*_{l_b}(\mathcal{D}) = \arg\max_{\mathbf{y}'\in\mathcal{Y}} |\{(\mathbf{x}, \mathbf{y}) \in \mathcal{D} \mid \mathbf{y} = \mathbf{y}'\}|$ when $l_b(p, v) = \mathbf{1}_{p\neq v}$

Data splitting

Test effects

- a test function f splits a data set into sub-data sets based on $f(\mathbf{X}_i)$
- notation $f(\mathcal{D}, o) = \{(\mathbf{x}, \mathbf{y}) \in \mathcal{D} \mid f(\mathbf{x}) = o\}$
- if f is the test at the root of tree T, then for each $\mathbf{x} \in f(\mathcal{D}, o)$, $g_T(\mathbf{x})$ is computed by the sub-tree associated to o

Tree growing operation



T $\Rightarrow$ $T + f$

$\boxed{\widehat{\mathbf{y}}_l^*(\mathcal{D})}$

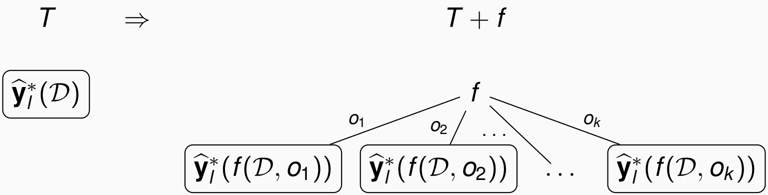
f

o_1 o_2 $\dots$ o_k

$\boxed{\widehat{\mathbf{y}}_l^*(f(\mathcal{D}, o_1))}$ $\boxed{\widehat{\mathbf{y}}_l^*(f(\mathcal{D}, o_2))}$ $\dots$ $\boxed{\widehat{\mathbf{y}}_l^*(f(\mathcal{D}, o_k))}$

Data splitting

Tree growing operation



$T \quad \Rightarrow \quad T+f$

$\boxed{\widehat{\mathbf{y}}_l^*(\mathcal{D})}$

f

o_1

o_2

$\dots$

o_k

$\boxed{\widehat{\mathbf{y}}_l^*(f(\mathcal{D},o_1))}$ $\boxed{\widehat{\mathbf{y}}_l^*(f(\mathcal{D},o_2))}$ $\dots$ $\boxed{\widehat{\mathbf{y}}_l^*(f(\mathcal{D},o_k))}$

- optimal growing for a given f
- empirical risk improvement

$$\widehat{R}_l(g_T,\mathcal{D}) - \widehat{R}_l(g_{T+f},\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left(l(\widehat{\mathbf{y}}_l^*(\mathcal{D}),\mathbf{y}) - l(\widehat{\mathbf{y}}_l^*(f(\mathcal{D},f(\mathbf{x}))),\mathbf{y})\right)$$

Judging a test/split

General principle

- is $T + f$ better than $T + f'$?
- empirical risk minimization:

$$T + f \text{ is better than } T + f' \Leftrightarrow \widehat{R}_l(g_{T+f}, \mathcal{D}) < \widehat{R}_l(g_{T+f'}, \mathcal{D})$$

Impurity measures

- when $|\mathcal{Y}| < \infty$ impurity measures are more adapted than risk based ones
- notation: $\widehat{p}(\mathbf{y}', \mathcal{D}) = \frac{1}{|\mathcal{D}|} \left|\{(\mathbf{x}, \mathbf{y}) \in \mathcal{D} \mid \mathbf{y} = \mathbf{y}'\}\right|$
- entropy: $\widehat{H}(\mathcal{D}) = -\sum_{\mathbf{y}\in\mathcal{Y}} \widehat{p}(\mathbf{y}, \mathcal{D}) \log \widehat{p}(\mathbf{y}, \mathcal{D})$
- Gini impurity: $I_G(\mathcal{D}) = \sum_{\mathbf{y}\in\mathcal{Y}} \sum_{\mathbf{y}'\in\mathcal{Y}\setminus\{\mathbf{y}\}} \widehat{p}(\mathbf{y}, \mathcal{D})\widehat{p}(\mathbf{y}', \mathcal{D})$

Impurity reduction

Definition

if $\mathcal{I}$ is an impurity measure, the impurity reduction induced by f is

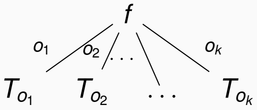
$$\Delta\mathcal{I}(f, \mathcal{D}) = \mathcal{I}(\mathcal{D}) - \sum_{o \in \mathcal{O}} \frac{|f(\mathcal{D}, o)|}{|\mathcal{D}|} \mathcal{I}(f(\mathcal{D}, o))$$

Remarks

- the empirical risk can be used as an impurity measure: the impurity reduction corresponds to the improvement in empirical risk!
- in ID3/C4.5 algorithms: $\mathcal{I} = \widehat{H}$ (information gain)
- in CART: $\mathcal{I} = I_G$

Growing algorithm

```
function GROWTREE($\mathcal{D}$, $l$, $\mathcal{I}$)
    if stopping conditions on $\mathcal{D}$ are fulfilled then
        return the tree $\boxed{\widehat{\mathbf{y}}_l^*(\mathcal{D})}$
    else
        select the best test $f$ according to impurity reduction based on $\mathcal{I}$
        for each possible output of $f$ $o \in \mathcal{O}$ do
            $T_o \leftarrow$ GROWTREE($f(\mathcal{D}, o)$, $l$, $\mathcal{I}$)
        end for
        return the tree
```



```
    end if
end function
```

Selecting the best test

ID3/C4.5

- work only for $|\mathcal{X}_j| < \infty$
- use only $f_n(\mathbf{x}) = x_j$
- the best f is selected by maximizing $\Delta\mathcal{I}(\mathbf{x} \mapsto x_j, \mathcal{D})$ over $j \in \{1, \ldots, P\}$

CART

- more complicated situation
- if $\mathcal{X}_j = \mathbb{R}$
  - sort the values $\{x_j \mid (\mathbf{x}, \mathbf{y}) \in \mathcal{D}\}$ into $(t_i)_{1 \leq i \leq N}$
  - compute all the mid-values, i.e. $\lambda_i = \frac{t_i + t_{i+1}}{2}$
  - select the best f (for $\mathcal{X}_j$) by maximizing $\Delta\mathcal{I}(\mathbf{1}_{x_j \leq \lambda_i}, \mathcal{D})$ over $i \in \{1, \ldots, N-1\}$

Selecting the best test

CART cont.

- if $|\mathcal{X}_j| < \infty$
  - comparing all the tests of the form $\mathbf{1}_{x_j \in A}$ in intractable ($2^{|\mathcal{X}_j|-1} - 1$ possibilities)
  - greedy approach
    - $u^{(1)} = \arg\max_{u \in \mathcal{X}_j} \Delta\mathcal{I}(\mathbf{1}_{x_j = u}, \mathcal{D})$
    - $u^{(2)} = \arg\max_{u \in \mathcal{X}_j \setminus \{u^{(1)}\}} \Delta\mathcal{I}\left(\mathbf{1}_{x_j \in \{u^{(1)}, u\}}, \mathcal{D}\right)$
    - …
    - $u^{(|\mathcal{X}_j|-1)} = \arg\max_{u \in \mathcal{X}_j \setminus \left\{u^{(1)}, \ldots, u^{(|\mathcal{X}_j|-2)}\right\}} \Delta\mathcal{I}\left(\mathbf{1}_{x_j \in \left\{u^{(1)}, \ldots, u^{(|\mathcal{X}_j|-2)}, u\right\}}, \mathcal{D}\right)$
    - select then the best subset from the $\left\{u^{(1)}, \ldots, u^{(k)}\right\}$ over k
  - the greedy approach considers only $\Theta\left(|\mathcal{X}_j|^2\right)$ subsets
  - faster optimal approaches in some particular cases (with $\Theta\left(|\mathcal{X}_j|\right)$ subsets)

Computational complexity

Impurity reduction

- naive approach: $\Theta(|\mathcal{D}|)$ for one test evaluation
- incremental approach: $\Theta(|\mathcal{D}|)$ for a set of tests in some cases

CART

- $\mathcal{X}_j = \mathbb{R}$
  - sorting is in $\Theta(N \log N)$
  - incremental evaluation of the impurity reduction is possible
  - $\Theta(N \log N)$ total
- $|\mathcal{X}_j| < \infty$
  - with $|\mathcal{Y}| = 2$ or $\mathcal{Y} = \mathbb{R}$ (with adapted impurity scores): $\Theta\left(|\mathcal{X}_j|\, N\right)$
  - more costly in arbitrary cases (e.g. $\Theta\left(|\mathcal{X}_j|^2 N^2\right)$
- global: if $|\mathcal{X}_j| \leq N$ and if the tree is balanced $\Theta(N(\log N)^2 P)$

Stopping the growth

Simple stopping conditions

- $|\mathcal{D}| \leq \mathrm{min}_{size}$: do not split small data sets
- if $|\mathcal{Y}| < \infty$, $\mathrm{max}_{\mathbf{y}\in\mathcal{Y}} \widehat{p}(\mathbf{y}, \mathcal{D}) \geq \mathrm{max}_{prob}$: do not split very pure data sets

Advanced stopping conditions

- $\mathrm{min}_{o\in\mathcal{O}} |f(\mathcal{D}, o)| \leq \mathrm{min}_{splitsize}$: do not split when the optimal test produces too small data sets
- $\frac{\Delta\mathcal{I}(f,\mathcal{D})}{\mathcal{I}(\mathcal{D})} \leq \mathrm{min}_{improv}$: do not split when the optimal test does not reduce enough the impurity

Missing data

Missing category

- missing observation for $\mathbf{x}_j \in \mathcal{X}_j$ with $|\mathcal{X}_j| < \infty$
- introduce a new value $\mathcal{X}'_j = \mathcal{X}_j \cap \{NA\}$ that encodes missing data
- proceed normally

Surrogate variables

- exploit local dependency between variables
- once the best test has been found rank all the other tests based on the misclassification error with respect to the output of the test
- store “in the node” a few tests (on distinct variables)
- when a value is missing, use alternative tests

Example

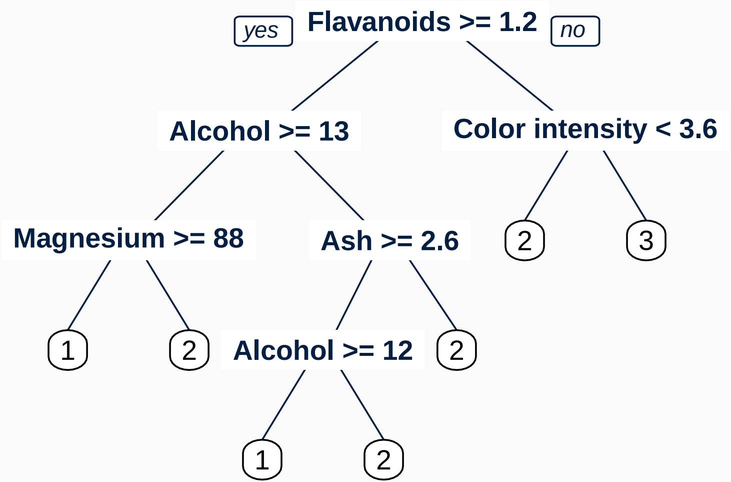
Data: wine

- chemical analysis of wines derived from three cultivars ($\mathcal{Y} = \{1, 2, 3\}$)
- 178 observations with 13 variables ($\mathcal{X} = \mathbb{R}^{13}$)
- $l_b(p, t) = \mathbf{1}_{p \neq t}$
- random split into a $\mathcal{D}$ for learning and $\mathcal{D}'$ for evaluation

Stopping criterion

- $\min_{size} = 2$
- $\min_{improv} = 0$

Example


yes
Flavanoids >= 1.2
no
Alcohol >= 13
Color intensity < 3.6
Magnesium >= 88
Ash >= 2.6
2
3
1
2
Alcohol >= 12
2
1
2

Quality

Confusion matrices

Learning set

	1	2	3
1	30	0	0
2	0	36	0
3	0	0	24

$\widehat{R}_{l_b}(g_T, \mathcal{D})) =$0

Validation set

	1	2	3
1	28	3	4
2	1	32	4
3	0	0	16

$\widehat{R}_{l_b}(g_T, \mathcal{D}') =$0.136

Unbalance

- notice that the data set is unbalanced
- this might explain the relatively poor performances on class 3

Adult data base

Data: Adult

- data extracted from the 1994 US Census
- 13 variables, most of them discrete (i.e. $|\mathcal{X}| < \infty$)
- classification problem: predict whether the annual income of a person is higher or lower than 50K US \$
- 32561 observations in $\mathcal{D}$ for learning
- 16281 observations in $\mathcal{D}'$ for validation/test
- $l_b(p, t) = \mathbf{1}_{p \neq t}$ + weights (re-balancing of the classes)

Results

Parameters

- $\min_{size} = 10$
- $\min_{improv} = 0$

Results

- enormous tree: 1678 leaves
- performances

Learning set

	<=50K	>50K
<=50K	21238	290
>50K	3482	7551

$\widehat{R}_{l_b}(g_T, \mathcal{D})) =$0.116

Validation set

	<=50K	>50K
<=50K	10018	855
>50K	2417	2991

$\widehat{R}_{l_b}(g_T, \mathcal{D}') =$0.201

- some overfitting

Smaller tree

Parameters

- $\min_{size} = 10$
- $\min_{improv} = 0.001$
- smaller tree: 20 leaves

Confusion matrices

Learning set

	<=50K	>50K
<=50K	19747	1161
>50K	4973	6680

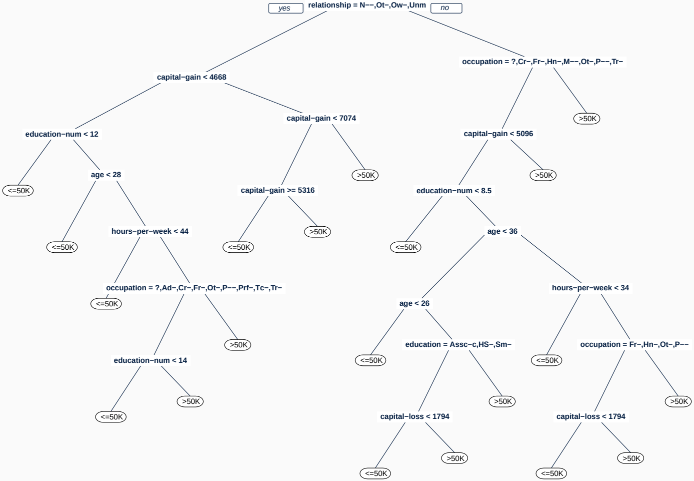
$\widehat{R}_{l_b}(g_T, \mathcal{D}))$ =0.188

Validation set

	<=50K	>50K
<=50K	9955	632
>50K	2480	3214

$\widehat{R}_{l_b}(g_T, \mathcal{D}')$ =0.191

The tree


yes
relationship = N−−,Ot−,Ow−,Unm
no
occupation = ?,Cr−,Fr−,Hn−,M−−,Ot−,P−−,Tr−
capital−gain < 4668
capital−gain < 7074
>50K
education−num < 12
capital−gain < 5096
age < 28
>50K
>50K
<=50K
capital−gain >= 5316
education−num < 8.5
hours−per−week < 44
>50K
age < 36
<=50K
<=50K
<=50K
occupation = ?,Ad−,Cr−,Fr−,Ot−,P−−,Prf−,Tc−,Tr−
hours−per−week < 34
<=50K
age < 26
>50K
education = Assc−c,HS−,Sm−
occupation = Fr−,Hn−,Ot−,P−−
education−num < 14
<=50K
<=50K
>50K
>50K
>50K
<=50K
capital−loss < 1794
capital−loss < 1794
>50K
>50K
<=50K
<=50K

Stopping criteria

From the experiments

- stopping parameters have a strong influence on the performances
- limited knowledge at the decision point:
  - the criteria operate only on a subset of the data
  - at best on a one step ahead basis
- large trees can overfit

Complexity control

- control the size of the tree
- but with a global point of view
- key idea:
  - grow the tree in order to reach a high quality on the training set (possibly with overfitting)
  - then prune the tree using another data set

Outline

Trees and decision trees

Growing a tree

Pruning a tree

Risk vs complexity

Tree complexity

- for a tree T, $|\{\mathbf{y} \in \mathcal{Y} \mid \exists \mathbf{x} \in \mathcal{X}, g_T(\mathbf{x}) = \mathbf{y}\}|$ is finite: it is the number of leaves in T!
- denote $|T|$ this number
- gives a good idea of the complexity of a tree in terms of g_T (rather than in terms of the structure of the tree)

Trade off

- when $|T|$ is large, $\widehat{R}_l(g_T, \mathcal{D})$ is small but the tree is complex and $\widehat{R}_l(g_T, \mathcal{D}')$ might be high
- optimize a compromise between those two aspects

$$C_{l,\alpha}(T, \mathcal{D}) = \widehat{R}_l(g_T, \mathcal{D}) + \alpha |T|$$

Pruning

Two problems

- given α, how to minimize $C_{l,\alpha}(T,\mathcal{D})$ over T in $\mathcal{T}(\mathcal{X},\mathcal{Y})$
- how to chose α

Pruning

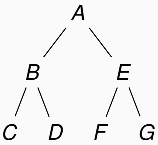
Two problems

- given α, how to minimize $C_{l,\alpha}(T, \mathcal{D})$ over T in $\mathcal{T}(\mathcal{X}, \mathcal{Y})$
- how to chose α

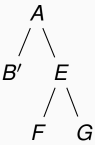
Pruning for $\mathcal{T}(\mathcal{X}, \mathcal{Y})$ exploration

- pruning a tree T consists in replacing any number of its sub-trees by leaves

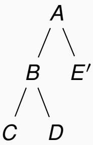
T



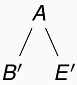
$T' \subset T$



$T'' \subset T$



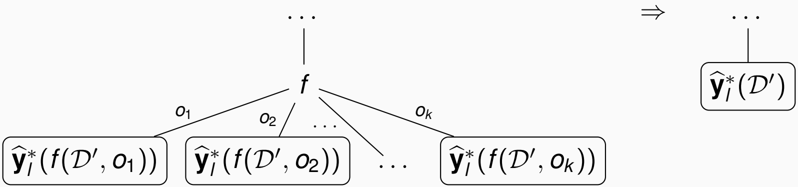
$T''' \subset T'$ and $T''' \subset T''$



Pruning cont.

Pruning decision trees

- best pruning assumption: the value of the leaf that replaces a sub-tree is optimal (risk-wise)
- contextual pruning: a data set is needed to compute the best value $\widehat{\mathbf{y}}_l^*(\mathcal{D})$
- a minimal pruning (when the sub-tree is of height 1) is the exact reverse operation of a growing step of the growing algorithm



$\ldots$
f
o_1
o_2
$\ldots$
o_k
$\widehat{\mathbf{y}}_l^*(f(\mathcal{D}', o_1))$
$\widehat{\mathbf{y}}_l^*(f(\mathcal{D}', o_2))$
$\ldots$
$\widehat{\mathbf{y}}_l^*(f(\mathcal{D}', o_k))$
$\Rightarrow$
$\ldots$
$\widehat{\mathbf{y}}_l^*(\mathcal{D}')$

Optimal tree

Strategy

- from a data set $\mathcal{D}$ grow a complex tree T_{max} (with strict stopping conditions)
- select the optimal tree by solving

$$T^* = \arg \min_{T \subset T_{\text{max}}} C_{l,\alpha}(T, \mathcal{D}) = \arg \min_{T \subset T_{\text{max}}} \left(\widehat{R}_l(g_T, \mathcal{D}) + \alpha \, |T| \right)$$

Optimal tree

Strategy

- from a data set $\mathcal{D}$ grow a complex tree $T_{\max}$ (with strict stopping conditions)
- select the optimal tree by solving

$$T^* = \arg\min_{T \subset T_{\max}} C_{l,\alpha}(T, \mathcal{D}) = \arg\min_{T \subset T_{\max}} \left(\widehat{R}_l(g_T, \mathcal{D}) + \alpha\,|T|\right)$$

Tractable?

- replacing $\mathcal{T}(\mathcal{X}, \mathcal{Y})$ by $\{T \in \mathcal{T}(\mathcal{X}, \mathcal{Y}) \mid T \subset T_{\max}\}$ simplifies tremendously the optimization problem
- but might still be a very large set!

Greedy pruning

Weakest node pruning

- given T consider all the nodes with only leaves as sub-trees
- compute for each such node the modification induced in $C_{l,\alpha}$ by pruning the node
- define $wnp(T)$ as the tree obtained by pruning the node which decreases the most $C_{l,\alpha}$

Greedy (but exact!) algorithm

- $T^{(0)} = T_{\max}$
- $T^{(k)} = wnp\left(T^{(k-1)}\right)$ (until $T^{(k-1)}$ becomes a leaf)
- $T^* = \arg\min_{T\in\left\{T^{(0)},\ldots,T^{(K)}\right\}} C_{l,\alpha}(T,\mathcal{D})$

Remarks

General

- $|T| \leq N$
- if $T' \subset T$, then $|T'| \leq |T| - 1$
- at most N trees to study
- notice than $T' \subset T$ does not imply any ordering between $C_{l,\alpha}(T, \mathcal{D})$ and $C_{l,\alpha}(T', \mathcal{D})$, thus the maximal decrease might actually be an increase

CART

- binary tree only
- pruning a node with only leaves as sub-trees reduces $|T|$ by exactly one
- wnp is determined only by the risk modification
- $\left\{T^{(0)}, \ldots, T^{(K)}\right\}$ does not depend on α

Compared to early stopping...

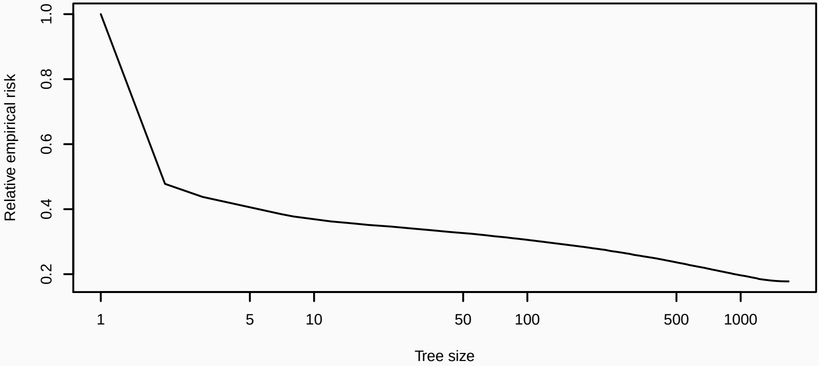
Alternative solution

- use different values of the stopping criterion to produce different trees
- pros and cons:
  - + simpler setting
  - + optimized implementation
  - - opaque parameters: predicting the size of the tree obtained with some parameters is impossible
  - - local decision only
  - - one step ahead only

Example

Adult data set

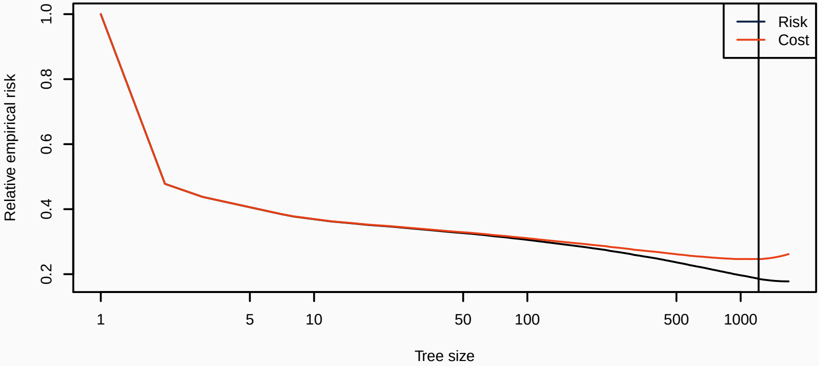
- $\min_{size} = 10$ and $\min_{improv} = 0$
- 1678 leaves



Example

Adult data set

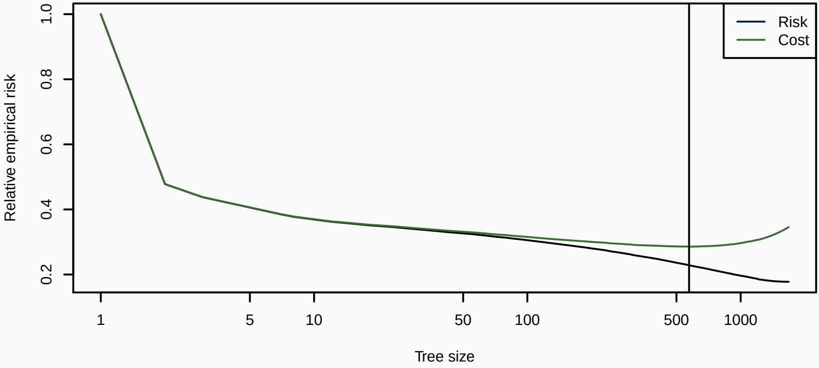
- $\text{min}_{size} = 10$ and $\text{min}_{improv} = 0$
- 1678 leaves



Example

Adult data set

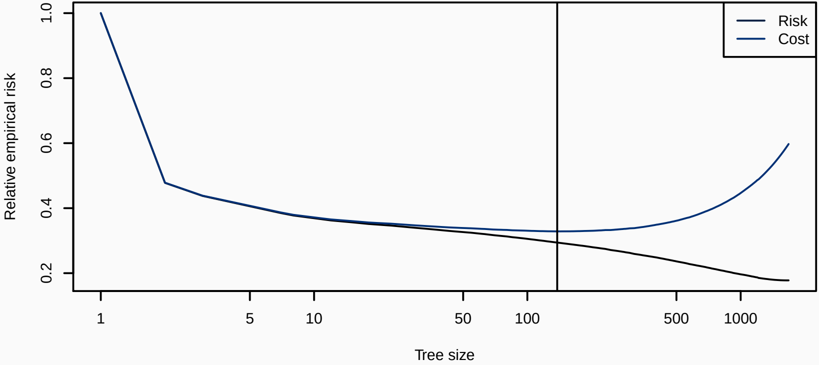
- $\min_{size} = 10$ and $\min_{improv} = 0$
- 1678 leaves



Example

Adult data set

- $\min_{size} = 10$ and $\min_{improv} = 0$
- 1678 leaves



Choosing α

Standard setting!

- the full CART algorithm is given:
  - a data set $\mathcal{D}$
  - a loss function l
  - an impurity measure $\mathcal{I}$
  - α
- and outputs $T^*(\mathcal{D}, l, \mathcal{I}, \alpha)$
- α is a parameter of the algorithm: must be optimized!

Validation approach

- build $T^*(\mathcal{D}, l, \mathcal{I}, \alpha)$ for some αs
- select the best α according to $\widehat{R}_l(g_{T^*(\mathcal{D},l,\mathcal{I},\alpha)}, \mathcal{D}')$

Interresting αs

α sensitivity (case of CART)

- weakest node pruning effect

$$C_{l,\alpha}(T,\mathcal{D}) - C_{l,\alpha}(wnp(T),\mathcal{D}) = \underbrace{\widehat{R}_l(g_T,\mathcal{D}) - \widehat{R}_l(g_{wnp(T)},\mathcal{D})}_{\text{independent of }\alpha} + \alpha$$

- α must be at least equal to $\widehat{R}_l(g_{wnp(T)},\mathcal{D}) - \widehat{R}_l(g_T,\mathcal{D})$ for $wnp(T)$ to be preferred to T
- one can compute intervals such that $T^*(\mathcal{D}, l, \alpha)$ depends only on the interval to which α belongs (cut off values)

Interresting αs

α sensitivity (case of CART)

- weakest node pruning effect

$$C_{l,\alpha}(T,\mathcal{D}) - C_{l,\alpha}(wnp(T),\mathcal{D}) = \underbrace{\widehat{R}_l(g_T,\mathcal{D}) - \widehat{R}_l(g_{wnp(T)},\mathcal{D})}_{\text{independent of }\alpha} + \alpha$$

- α must be at least equal to $\widehat{R}_l(g_{wnp(T)},\mathcal{D}) - \widehat{R}_l(g_T,\mathcal{D})$ for $wnp(T)$ to be preferred to T
- one can compute intervals such that $T^*(\mathcal{D}, l, \alpha)$ depends only on the interval to which α belongs (cut off values)

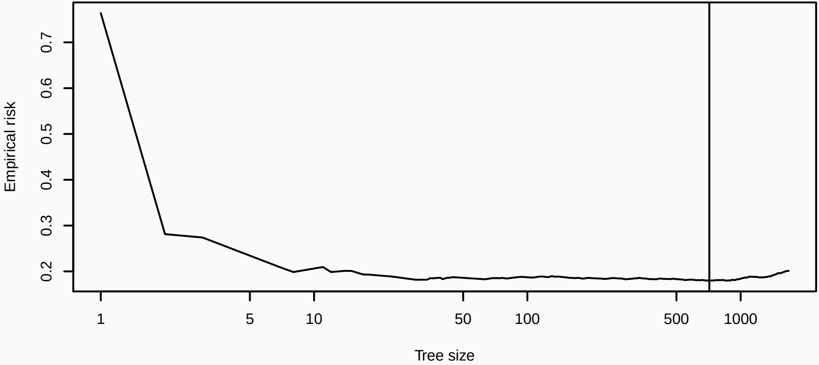
In practice

- in the validation setting, only $\{T^{(0)}, \ldots, T^{(K)}\}$ (does not depend on α)
- in resampling approaches using the intervals is important

Example

Adult data set

- $\mathrm{min}_{size} = 10$ and $\mathrm{min}_{improv} = 0$
- from 1678 leaves to 713



Example

With cross-validation

- 138 leaves
- confusion matrices

Learning set

	<=50K	>50K
<=50K	20164	861
>50K	4556	6980

$\widehat{R}_{l_b}(g_T, \mathcal{D}))$ =0.166

Validation set

	<=50K	>50K
<=50K	9973	612
>50K	2462	3234

$\widehat{R}_{l_b}(g_T, \mathcal{D}')$ =0.189

- full tree

Learning set

	<=50K	>50K
<=50K	21238	290
>50K	3482	7551

$\widehat{R}_{l_b}(g_T, \mathcal{D}))$ =0.116

Validation set

	<=50K	>50K
<=50K	10018	855
>50K	2417	2991

$\widehat{R}_{l_b}(g_T, \mathcal{D}')$ =0.201

Example

With cross-validation

- 138 leaves
- confusion matrices

Learning set

	$<$=50K	$>$50K
$<$=50K	20164	861
$>$50K	4556	6980

$\widehat{R}_{l_b}(g_T, \mathcal{D}))$ =0.166

Validation set

	$<$=50K	$>$50K
$<$=50K	9973	612
$>$50K	2462	3234

$\widehat{R}_{l_b}(g_T, \mathcal{D}')$ =0.189

- small tree (20 leaves)

Learning set

	$<$=50K	$>$50K
$<$=50K	19747	1161
$>$50K	4973	6680

$\widehat{R}_{l_b}(g_T, \mathcal{D}))$ =0.188

Validation set

	$<$=50K	$>$50K
$<$=50K	9955	632
$>$50K	2480	3214

$\widehat{R}_{l_b}(g_T, \mathcal{D}')$ =0.191

CART in summary

```
function CART($\mathcal{D}$, $I$, $\mathcal{I}$)
    $T_{\text{max}} \leftarrow$ GrowTree($\mathcal{D}$, $I$, $\mathcal{I}$)
    $T \leftarrow T_{\text{max}}$
    $Ts \leftarrow \{T_{\text{max}}\}$
    $\alpha s \leftarrow \{\}$
    while $T$ is not a leaf do
        $T \leftarrow wnp(T)$
        $Ts \leftarrow Ts \cup \{T\}$
        compute $\alpha$ the next cut off value
        $\alpha s \leftarrow \alpha s \cup \{\alpha\}$
    end while
    return $(Ts, \alpha s)$
end function
```

And then...

- use a validation technique to select to best tree from *Ts*
- or the best α from αs

Outline

Trees and decision trees

Growing a tree

Pruning a tree

Summary

Decision trees

- simple program like models for machine learning
- pros and cons
  - + relatively fast ($\Theta(N(\log N)^2P)$)
  - + adapted to mixed data
  - + handle missing data
  - + interpretable (for small trees)
  - - medium performances
  - - useful only when interactions between variables are important
  - - highly irregular model (especially for $\mathcal{Y} = \mathbb{R}$)
  - - high variance

Summary

Decision trees

- simple program like models for machine learning
- pros and cons
  - + relatively fast ($\Theta(N(\log N)^2 P)$)
  - + adapted to mixed data
  - + handle missing data
  - + interpretable (for small trees)
  - - medium performances
  - - useful only when interactions between variables are important
  - - highly irregular model (especially for $\mathcal{Y} = \mathbb{R}$)
  - - high variance

Combining trees

- trees as building blocks for better algorithms
- bagging, random forests and boosting

Licence

Version

Last git commit: 2018-06-06
By: Fabrice Rossi (Fabrice.Rossi@apiacoa.org)
Git hash: 1b39c1bacfc1b07f96d689db230b2586549a62d4