Machine learning 2
Decision trees

Pierre Chainais

centralelille
ÉCOLE CENTRALE DE LILLE

Trajectory

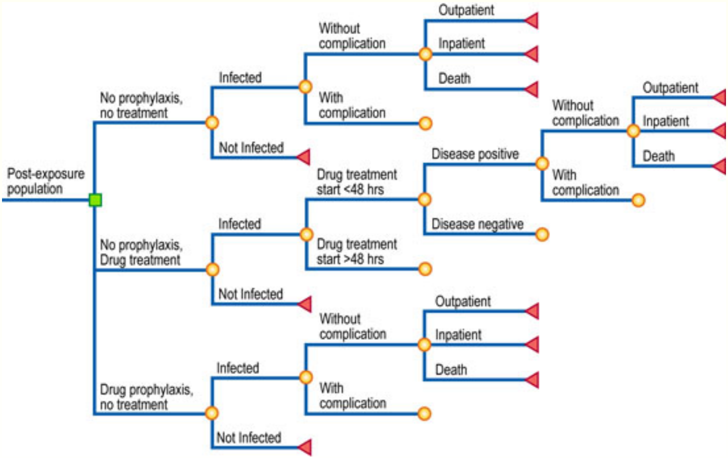
1. Decision trees
   - Motivations
   - Principes de construction
   - Binary features / others
   - Descending inference of a decision tree
   - Pruning
   - Properties: advantages, limitations, generalizations
   - Bagging
   - Random forests
   - Complements

References

- Classification and regression trees. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Chapman & Hall, 1984.
- Pattern classification. R. Duda, P. Hart and D. Stork. Wiley, New York, 2000.
- Bagging Predictors. L. Breiman. Machine Learning 26:123-140, 1996.
- Random Forests. L. Breiman. Machine Learning 45:5-32, 2001.

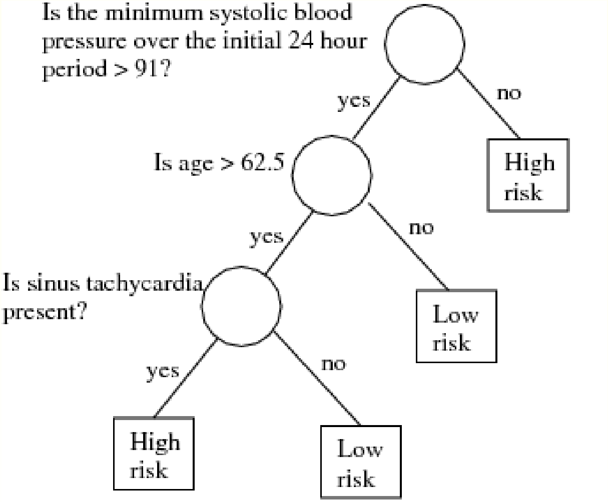
Motivations

A usual approach



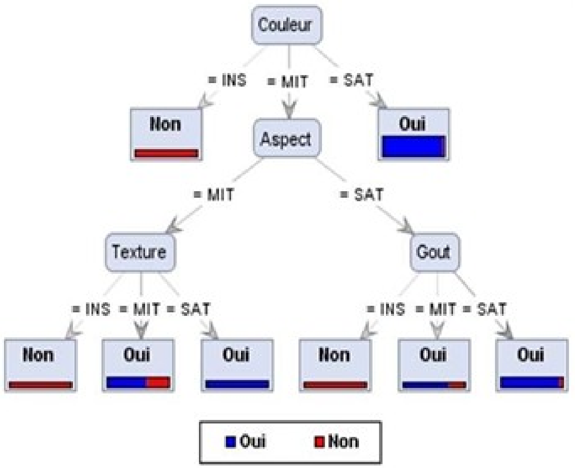
Motivations

A usual approach



Motivations

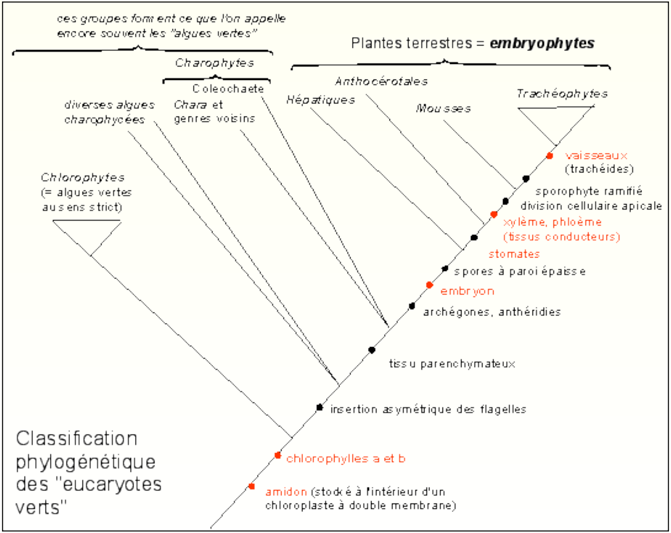
A usual approach



Arbre de décision

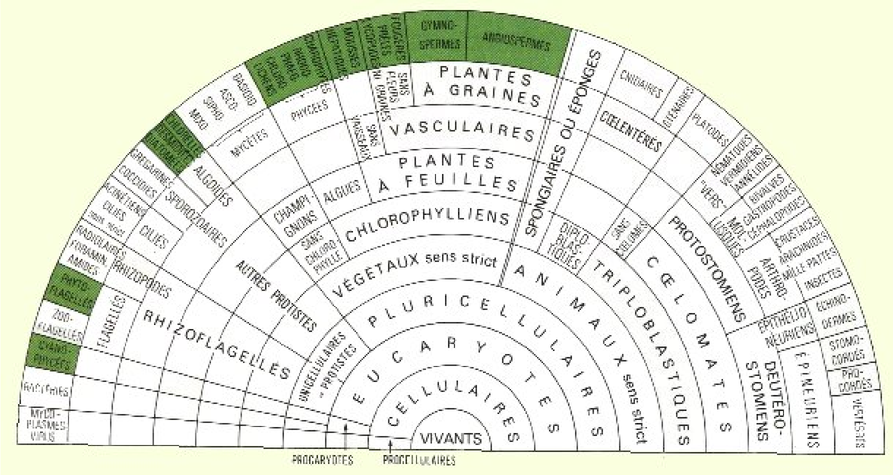
Motivations

A usual approach



Motivations

A usual approach



Principle of construction

Elementary tree for binary decision

- Binary feature $X_i \in \{0,1\}$, $Y_i \in \{0,1\}$
- Training set (X_i, Y_i), $i = 1, \ldots, n$
- Table of contingence

Y / X	0	1	Total
0	$n_{0\|0}$	$n_{0\|1}$	n_0
1	$n_{1\|0}$	$n_{1\|1}$	n_1
Total	$n_{:\|0}$	$n_{:\|1}$	n

- $\Pr(Y = k | X = \ell) = \pi_{k|\ell}$ avec $\pi_{0|\ell} + \pi_{1|\ell} = 1$
- Maximum likelihood

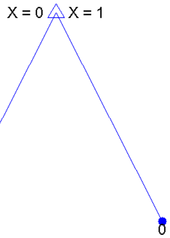
$$\widehat{\pi}_{k|\ell} = \frac{n_{k|\ell}}{n_{0|\ell} + n_{1|\ell}}$$

- Decision for the 0/1 loss function

$$f(x) = \begin{cases} 1 & \text{if } n_{1|x} > n_{0|x} \\ 0 & \text{else} \end{cases}$$

Principle of construction

Elementary tree for binary decision



Principle of construction

- Numerous predictors (D = high dimension)
- Quantitative or qualitative features
- Impossible to consider all possible values of $\mathbf{x} = (x_1, \ldots, x_D)$

$\Longrightarrow$ Classification, decision trees

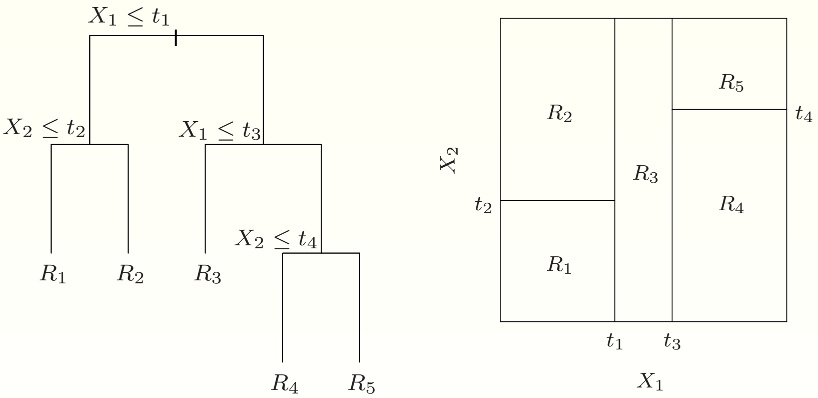
How to learn...

- the good questions? (selectors)
- the good answers? (decision)
- to keep only useful questions? (pruning)

Principle of construction

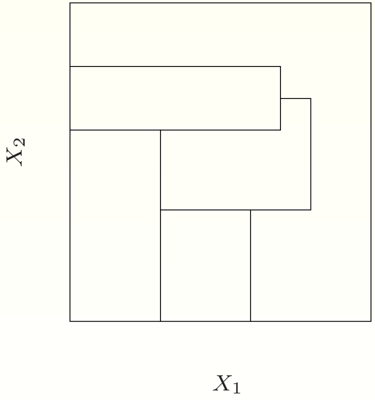
- (X_i) $i = 1, \ldots, n$: training set
- classification tree= successive binary partitions of (X_i), starting from the full training set
- union of regions occupied by 2 daughter nodes
  $=$ region occupied by the father node
- each terminal node $\Longrightarrow$ classification rule
- prediction $=$ terminal node for X

Principle of construction

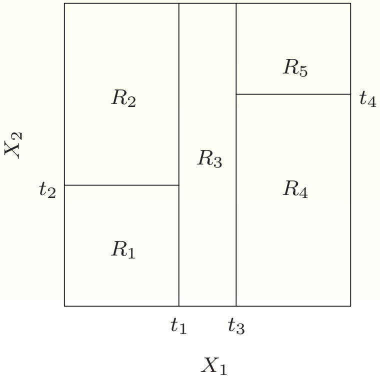


Principle of construction

Remark: no hierarchical partition.



$\neq$



Binary features / others

- Vector $X = (X_1, \ldots, X_p)$
- Categorical or quantitative features
- Each partition of the data at some node uses one variable only
- If $X_j \in \{1, \ldots, M\}$, question of the form "$X_j \in A$?", $A \subset \{1, \ldots, M\}$ (e.g. M binary questions)
- If $X_j \in \mathbb{R}$, since the training set is finite, there exists a finite number of questions "$X_j \leq c$?"

Recursive construction of a decision tree

Procedure *Construct-tree (node m)*

begin

if *All the points of node m nelong to the same class*

then Create a new leaf with this class

else

Choose the best feature to create a node

Test this feature to separate m in two daughter nodes m_L and m_R

`Construct-tree` (m_L)

`Construct-tree` (m_R)

end if

end

Algorithm

3 essential elements:

- Choice of the criterion to partition this node
- Choice of the decision to take at terminal node (leaf)
- Choice of a partition rule at a node

Impurity measure of a node

- R_m = set of N_m training samples at node m,

$$p_{m,k} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

  the frequency of each classe k at node m

- $d(m) = \arg\max_k p_{m,k}$ le "majority vote" at node m

How to measure the quality of a partition rule at node m?

Impurity measure of a node

How to measure the quality of a partition rule at node m?

$$p_{m,k} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{I}(y_i = k)$$

Impurity function :
ϕ defined on $(p_1, \ldots, p_K)$ with $p_k \geq 0$ et $\sum_{k=1}^{K} p_k = 1$ such that

- ϕ has a unique maximum at $(\frac{1}{K}, \ldots, \frac{1}{K})$
- ϕ is minimal at points $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$, $\ldots$
- $\phi(p_1, \ldots, p_K) = \phi(p_{\sigma(1)}, \ldots, p_{\sigma(K)})$ for any permutation σ

Impurity measure

$$p_{m,k} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{I}(y_i = k)$$

- Standard impurity measures
  - Classification error

$$\frac{1}{N_m} \sum_{i \in R_m} \mathbb{I}(y_i \neq d(m)) = 1 - p_{m,d(m)}$$

  - Gini's index

$$\sum_{k \neq k'} p_{m,k} p_{m,k'} = \sum_{k=1}^{K} p_{m,k}(1 - p_{m,k})$$

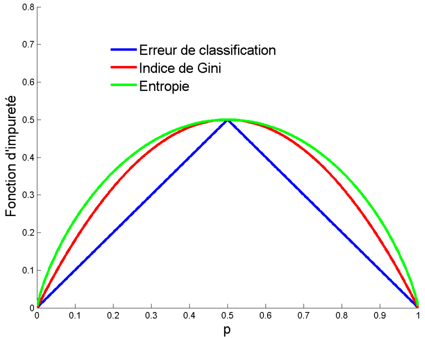
  - Cross-entropy or deviance

$$-\sum_{k=1}^{K} p_{m,k} \log p_{m,k}$$

Impurity measure

Simple case: K=2

- If $K = 2$ and p is the proportion of the 2nd class
  - Classification error : $1 - \max(p, 1-p)$
  - Gini's index : $2p(1-p)$
  - Cross-entropy or deviance : $-p \log p - (1-p) \log(1-p)$



Impurity measure and choice of the selector

- **partition** s = choice of a variable x_i + best cut w.r.t. x_i
- **the quality of a partition** s at node m is measured by

$$\Delta\phi(s, m) = \phi(p_m) - (\pi_L \phi(p_{m_L}) + \pi_R \phi(p_{m_R}))$$

  where π_L and π_R are the proportions of data in resp. the left and right nodes.
- m_L and m_R are daughter nodes of node m
- $\phi(p_{m_L})$ and $\phi(p_{m_R})$ are the impurity measures at m_L et m_R
- $\phi(p_m)$ = impurity of the father node,
- feasible in an exhaustive manner for reasonable data sets.

Pre-pruning: stopping criterion

- Intuitive criterion: stop at node m when

$$\arg\max_s \Delta\phi(s, m) < \varepsilon$$

where ε is some threshold
$\Rightarrow$ often not very efficient: several successive partitions which are not efficient by themselves can lead to an important gain.

- **Best strategy**
  1. Construct a deep tree T_0 and stop when only a minimum number of data remains at each node (e.g. 5 elements)
  2. Cut branches according to some complexity criterion

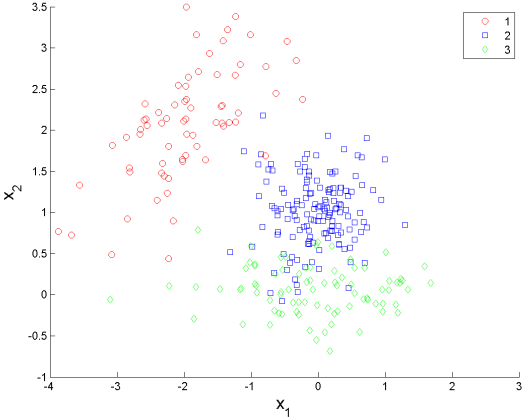
Pruning: regularization

- One can define the cost-complexity criterion

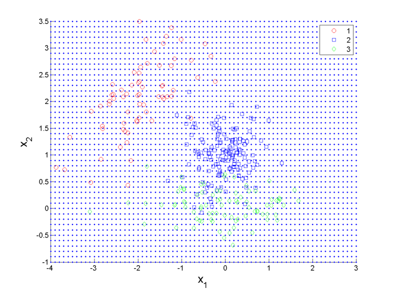
$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m \phi(p_m) + \alpha |T|$$

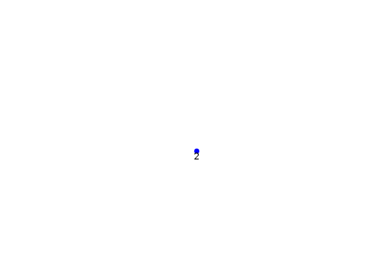
  where $|T|$ is the number of terminal node of T and $m = 1, \ldots, |T|$ correspond to terminal nodes
- The parameter α tunes the complexity of the tree
  - $\alpha = 0$ corresponds to T_0
  - $\alpha \gg 1$ will favour shallow trees
- α can be estimated using CV.
- For fixed α, find the sub-tree $T_\alpha \subseteq T_0$ minimizing $C_\alpha(T)$
- There exists a unique sub-tree minimizing $C_\alpha(T)$ (provable)

Example: synthetic data set

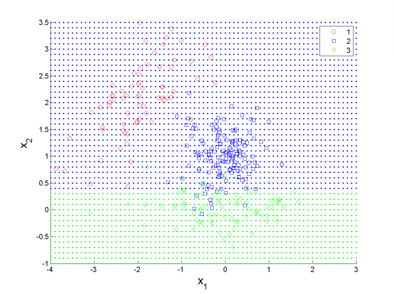


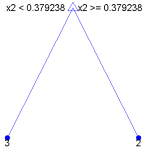
Example: synthetic data set



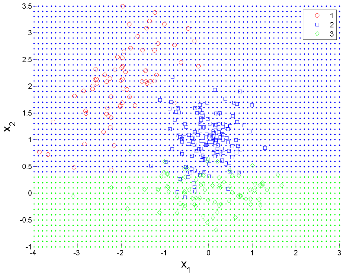


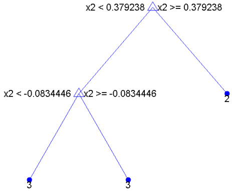
Example: synthetic data set



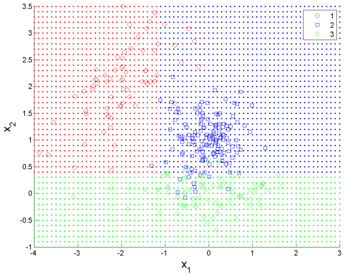


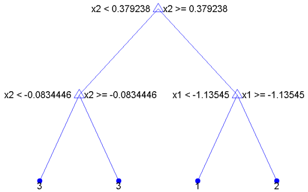
Example: synthetic data set



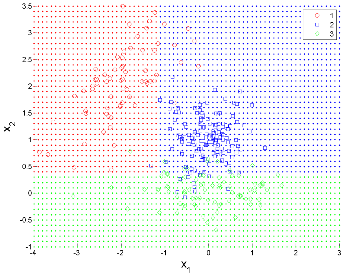


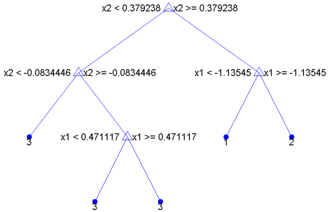
Example: synthetic data set



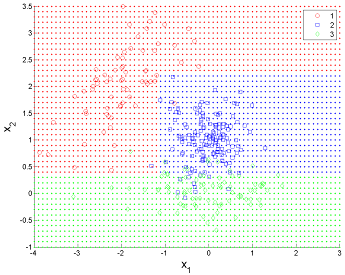


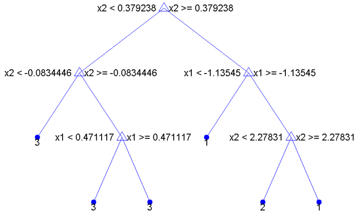
Example: synthetic data set



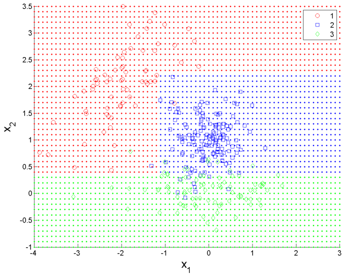


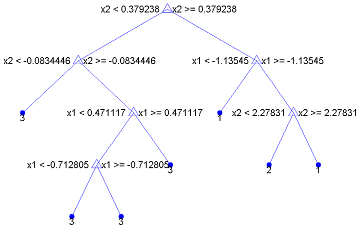
Example: synthetic data set



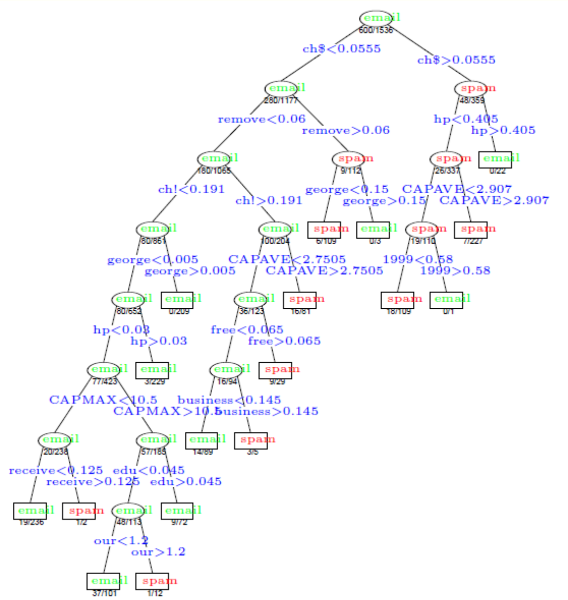


Example: synthetic data set





Example: spam filtering



- **Advantages** : scaling to high dimensions (D tests), $K \geq 2$ classes, any variables...
- **Binary partitions?**
  - Partitioning in more than 2 daughter nodes...
  - ... less efficient due to a faster fragmentation of the data
- **Linear combinations**
  - Partitions according to a linear model $\sum a_j X_j \leq s$
  - Weights a_j can be estimated unsing optimization
- **Instability of trees**
  - A small change in the training set may completely change the tree...
  - ... due to the recursive construction
  - Solution : *bagging* and *random forests* (Breiman 1996, 2001)

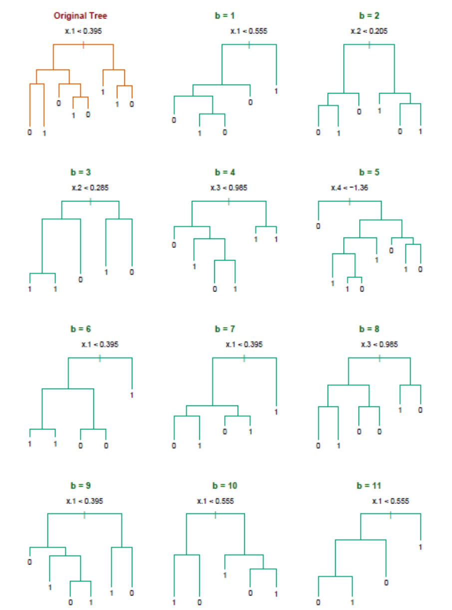
Bagging : Bootstrap AGGregatING

- **Idea** :
  artificially create several data sets $\mathcal{X}_b$, $b = 1, \ldots, B$
- **Bootstrap** :
  sample $\mathcal{X}_b$ with replacement from the training set
- **Aggregating** :
  - a classification tree for each $\mathcal{X}_b$,
  - prediction : let $\widehat{y}_b(x)$ the prediction from each tree b then

$$m_k(x) = \sum_{b=1}^{D} I(\widehat{y}_b(x) = k)$$

$$\widehat{y}(x) = \arg\max m_k(x)$$

Bagging


Original Tree
x.1 < 0.395
b = 1
x.1 < 0.555
b = 2
x.2 < 0.205
b = 3
x.2 < 0.285
b = 4
x.3 < 0.985
b = 5
x.4 < −1.36
b = 6
x.1 < 0.395
b = 7
x.1 < 0.395
b = 8
x.3 < 0.985
b = 9
x.1 < 0.395
b = 10
x.1 < 0.555
b = 11
x.1 < 0.555

Bagging

- \- No more hierarchical structure
- \+ Improves performances (classification error) and less unstable
- ▶ Random forests (Breiman, 2001): modification of bagging to decorrelate trees

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

Complements

- Classical algorithms: CART, ID3, C4.5 and C5.0...
- Usual in data mining (fouille de données)
- Boosting + elementary "stump" tree
- Even better: boosting trees (gradient boosting...)
- Regression trees: minimizing the mean square error criterion (in place of impurity)
- Softwares & toolboxes : scikit-learn, Matlab, R, Weka 3, See5/C5.0, Java...