Machine/Statistical Learning

Supervised learning
Decision Trees, Ensemble learning

Florent Chatelain, Olivier.JJ.Michel

Grenoble-INP, GISAP-Lab

ENSTA, feb. 2020

Material used, references

- www Sources for some figures & examples
  https ://scikit-learn.org/stable/
  https ://dimensionless.in/introduction-to-random-forest/
  https ://gluon.mxnet.io/index.html
  https ://skymind.ai/wiki/
- Books used
  K. P. Murphy : *"Machine Learning : A probabilistic perspective"*, MIT Press, 2012
  A. C. Azencott :*"Introduction au Machine Learning"*, Dunod, 2018
  A. Geron :*"Hands-On Machine Learning with Scikit-Learn, Keras, a* O'Reilly Media, 2019

See TREES_iris_data_example.ipynb
See TREES_sonar_data_example.ipynb

Decision trees

Trees are versatile hierarchical ML algorithms able to perform both classification and regression tasks. decision trees

- allow to deal with discrete (as well as continuous) features or attributes (shape, size, color,...)
- tackle Multi-class classification problems, without going back to binary formulation
- tackle the problem of multi-modal classes (different criteria apply in different regions of the observation space)

Remind : as we deal with supervised approach, knowledge of a training set on $\mathcal{X} \times Y$ is assumed : $\mathcal{T} = \{(x_i, y_i), i = 1 \ldots, N\}$

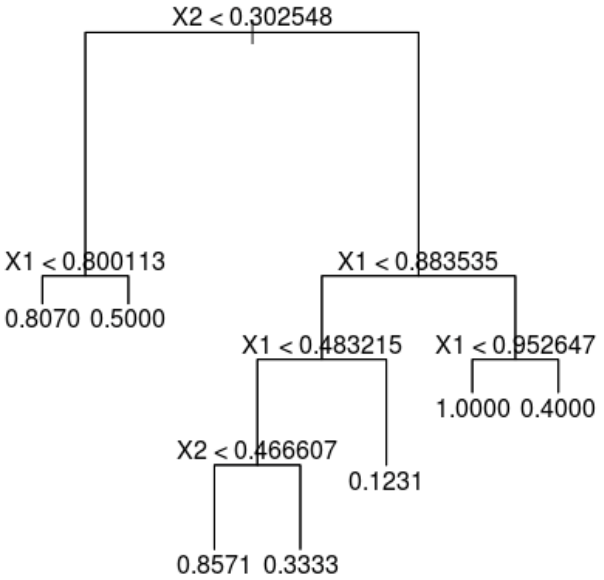
Decision trees definition

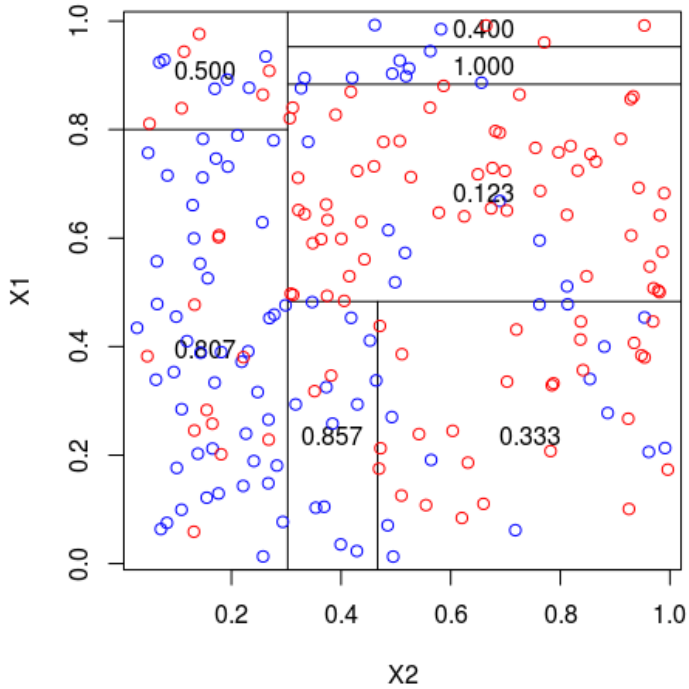
A decision tree is a prediction model using sequential conditional operations, organized in a tree :

- each node corresponds to a (binary) test on a single variable
- each child of a parent node corresponds to a possible answer of the test
- each leave or terminal node is associated to a single label or output value.

Let $\{\mathcal{R}_1, \ldots, \mathcal{R}_{|T|}\}$ be the set of regions describing the partition induced by the tree on the observation space $\mathcal{X}$

Example





Analytical formulation of the decision function :

$$f(x) = \sum_{k=1}^{|T|} \mathbb{I}[x \in \mathcal{R}_k] g(\mathcal{R}_k)$$

where
$$\begin{array}{ll} g(\mathcal{R}_k) \arg\max_c \sum_{x \in \mathcal{R}_k} \delta(y, c) & \text{for classification} \\ g(\mathcal{R}_k) = \frac{1}{|\mathcal{R}_k|} \sum_{x \in \mathcal{R}_k} y & \text{for regression} \end{array}$$

Growing trees

CART Construction (Breiman 1984) : built a deterministic data structure for modeling decision rules for a specific classification or regression problem. The aim is to create a model that predicts the value of a target variable based on several input variables.

→ Each "interior" node of a tree corresponds to one of the input variables ; Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

→ A tree can be "learned" by splitting the source set into subsets based on an attribute value test. The "recursion" is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions (greedy algo).

→ As any multiway split may be summerised as a series of binary splits, the focus is put on binary splits.

→ Splits are obtained by choosing a test at each step that "best" splits the set of items : What does mean Choosing the "best" split $S = S_r \cup S_l$ such that $S_r \cup S_l = \emptyset$ for the test property T ? An obvious heuristic is to choose the the query that decreases the impurity a much as possible.

Growing trees -cont'd

Splitting criterion for regression

MSE in general : look for the coordinate index j and the splitting point s satisfying

$$\arg\min_{(j,s)} \left(\sum_{x_i \in S_r(j,s)} (y_i - y_r(j,s))^2 + \sum_{x_i \in S_l(j,s)} (y_i - y_l(j,s))^2 \right)$$

where $y_r(j,s)$ resp $y_l(j,s)$) are the output associated to S_r resp. S_l.

Splitting criterion for classification

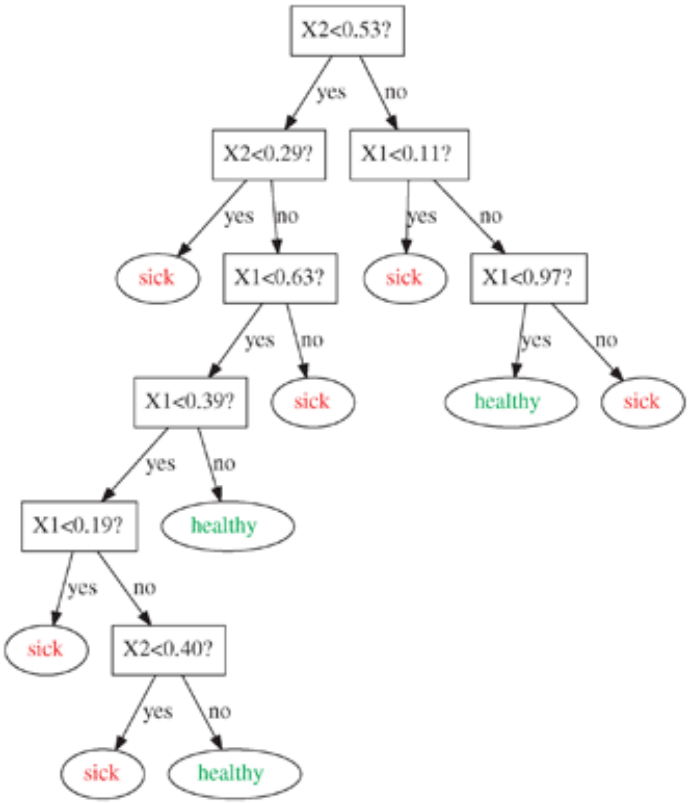
Impurity based criterion : look for the coordinate index j and the splitting point s satisfying

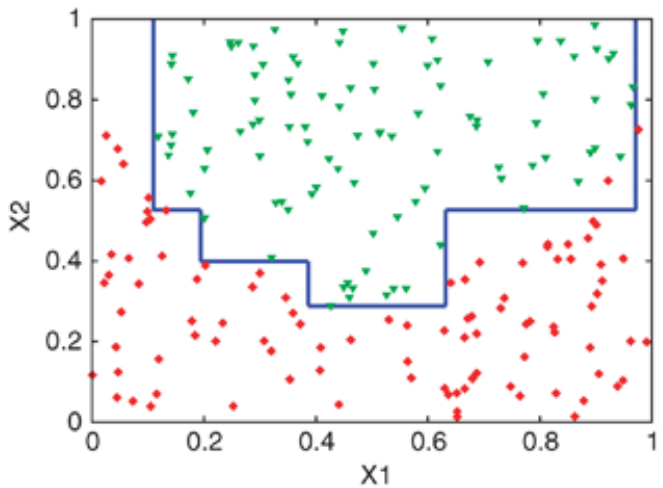
$$\arg\min_{(j,s)} \left(\frac{|S_r(j,s)|}{n} \text{Imp}(S_r(j,s)) + \frac{|S_l(j,s)|}{n} \text{Imp}(S_l(j,s)) \right)$$

where Imp() is the Impurity function and $n = |S| = |S_r| + |S_r|$.

Consequence

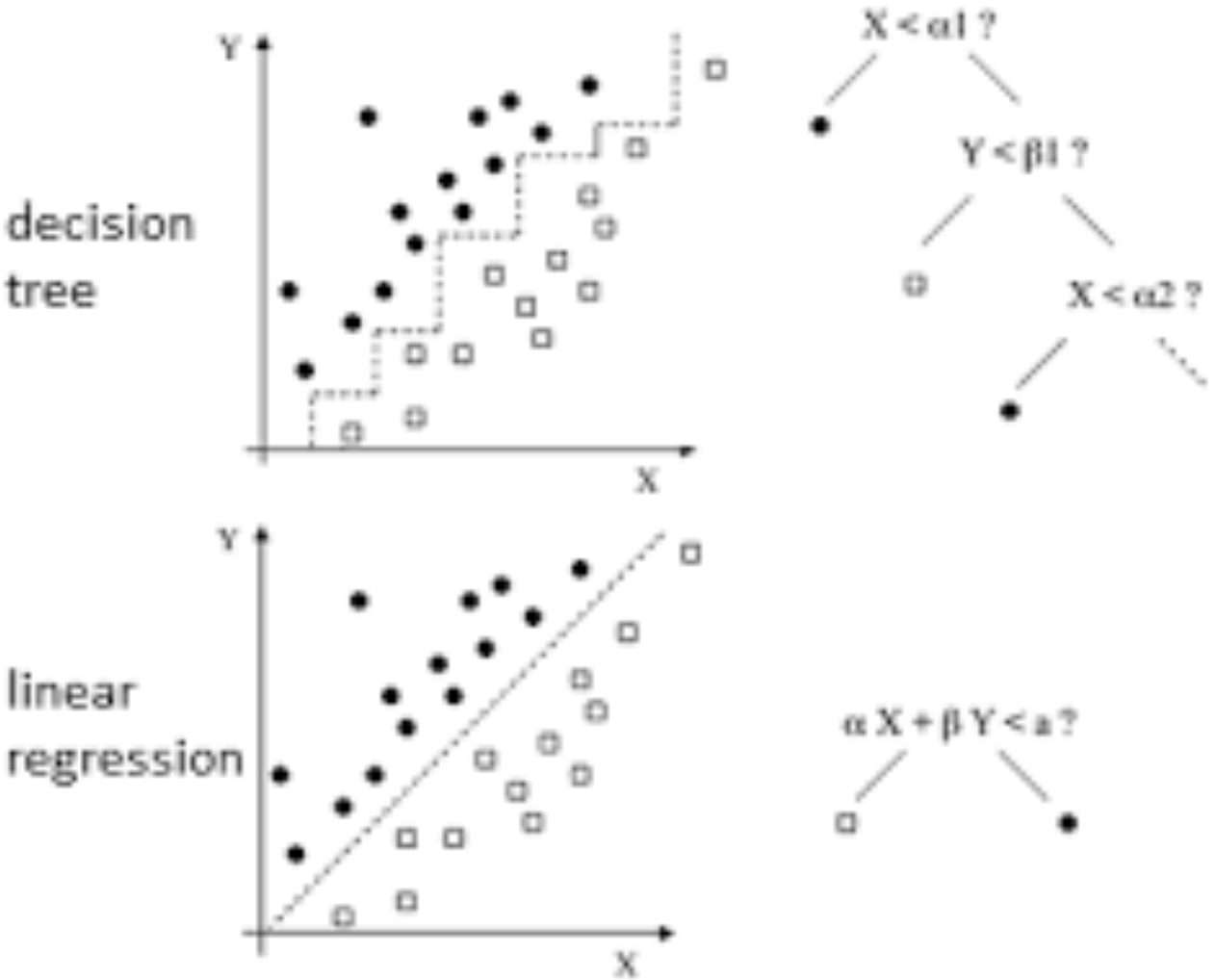
Each split leads to a straight line classifying the dataset into two parts.
Thus, the final decision boundary will consist of straight lines (or boxes).





Consequence

In comparison to regression, a decision tree can fit a stair case boundary to classify data.



Impurity function for classification trees

Assume that $\forall (x_i, y_i) \in \mathcal{T}, y_i \in \{1, \dots C\}$
Let $p_c \overset{def}{=} \frac{1}{|S|} \sum_{x_i \in S} \delta(y_i, c)$; Widely used Impurity functions are

- Gini impurity
- Entropy (or information)
- Misclassification

rk : Split wrt a single attribute. In general for non numerical attributes, an exhaustive search over all possibilities is performed. For real values attributes, gradient method for identifying a separating hyperplane may be used. **However, simple threshold on a single attribute is often preferred**.

Gini impurity

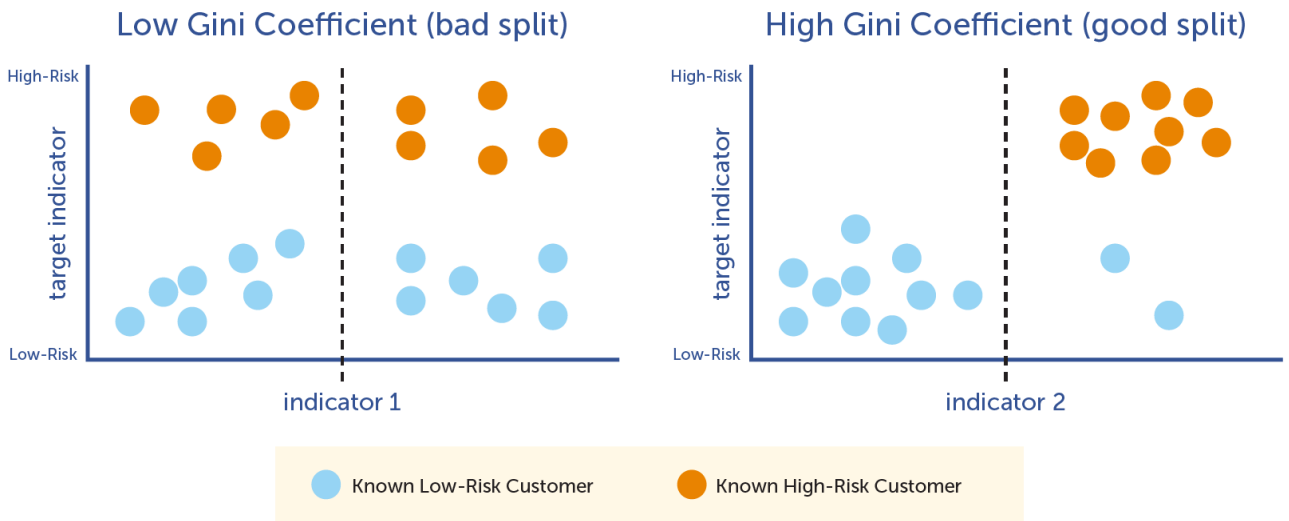
Gini Impurity of a set **S** of cardinal n measures the probability that a randomly chosen element of the set would be incorrectly labeled if randomly labeled according to the label distibution in **S**.

$$GI(S) = \sum_{c=1}^{C} P(y=c).P(y \neq c) = \sum_{c=1}^{C} p_c[1-p_c] = 1 - \sum_{c=1}^{C} p_c^2$$

then

$$\Delta GI(S) = GI(S) - P(S_l)GI(S_l) - (1-P(S_l))GI(S_r)$$

where $P(S_l) = \frac{n_l}{n}$ and $n_l + n_r = n$, $n = |S|$



Entropy Impurity

$$H(S) = -\sum_{k=1}^{C} P(y=k)\log_2 P(y=k)$$

the the information gain (IG) associated to the split (or partition) (S_l, S_r) is

$$IG(S) = H(S) - H(S|(S_l, S_r)) = H(S) - P(S_l)H(S_l) - P(S_r)H(S_r)$$

where $H(S_l)$ (resp $H(S_r)$) are evaluated by using the empirical probabilities of the classes estimated from the subset S_l (resp S_r) :

$$H(S_l) = -\sum_{k=1}^{C} P(y=k|X \in S_l)\log_2 P(y=k|X \in S_l)$$

Misclassification Index

$$MI(N) = 1 - \max_c P(y = c)$$

and hence the gain in MI

$$\begin{aligned} \Delta MI(N) &= MI(N) - P(S_l)MI(N_l) - P(S_r)MI(N_r) \\ &= MI(N) - 1 + P(S_l)\max_k P(y = k|X \in S_l) + \\ &\quad P(S_r)\max_k P(y = k|X \in S_r) \end{aligned}$$

Partial conclusion on decision tree growing

Tree Pruning

When should we stop splitting ?

- At a prescribed depth in the tree
- When S is 'pure'
- When $n = |S|$ is too small to warrant statistical significance of the estimated impurity

or grow the tree as long as possible and then operate Cost complexity pruning

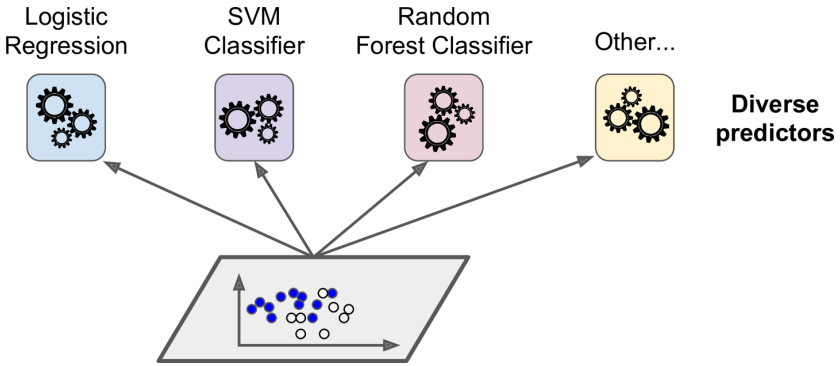
$$\mathcal{C}_\lambda(\mathcal{R}_1, \ldots, \mathcal{R}_{|T|}) = \sum_{t=1}^{|\mathcal{T}|} n_t \text{Imp}(\mathcal{R}_t) + \lambda |T|$$

BUT in PRACTICE Trees are weak learners
(oversimple models, low performances, sensitivity to outliers...)

$\Rightarrow$ the solution is ENSEMBLE LEARNING : "the wisdom of crowds"

Ensemble learning, bagging

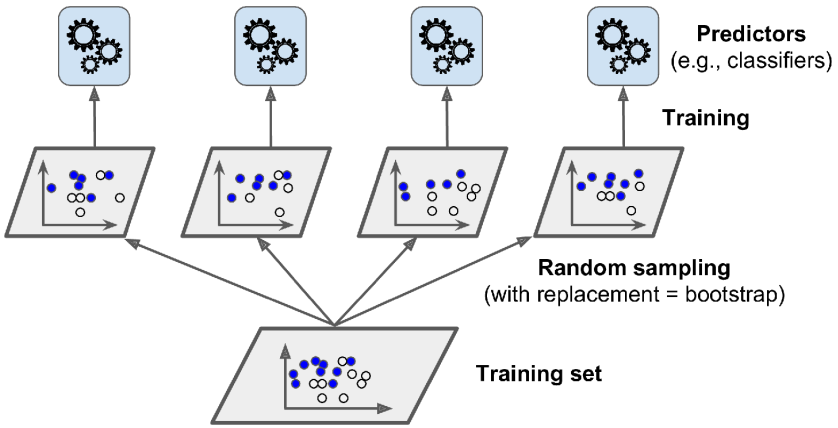
Multiple approaches or algorithms :



Ensemble Prediction : aggregate all results of multiple weak learners by

- weighted average (regression)
- majority vote
- ...

Bagging, with many instances of a single algorithm :



Bagging : Bootstrap Aggregating (Breiman 1996)

Motivations :

- Combining weak learners leads to decrease the variance of the predictor
- Each of the B weak learners uses a sample of m samples drawn randomly in $\mathcal{T}$
- Weak learners may be computed in parallel

rk1 : Ensemble methods work best when the predictors are as independent from one another as possible. One way to get diverse classifiers is to train them using very different sample sets from the training set .

- Sampling with replacement $\rightarrow$ Bagging
- Sampling withoutreplacement $\rightarrow$ Pasting

rk2 Alternative sampling :

- The features may be sampled (e.g. if $\mathcal{X}$ is high dimensional) $\rightarrow$ Subspace sampling
- Both instances from $\mathcal{T}$ and features may be sampled $\rightarrow$ Random patches method

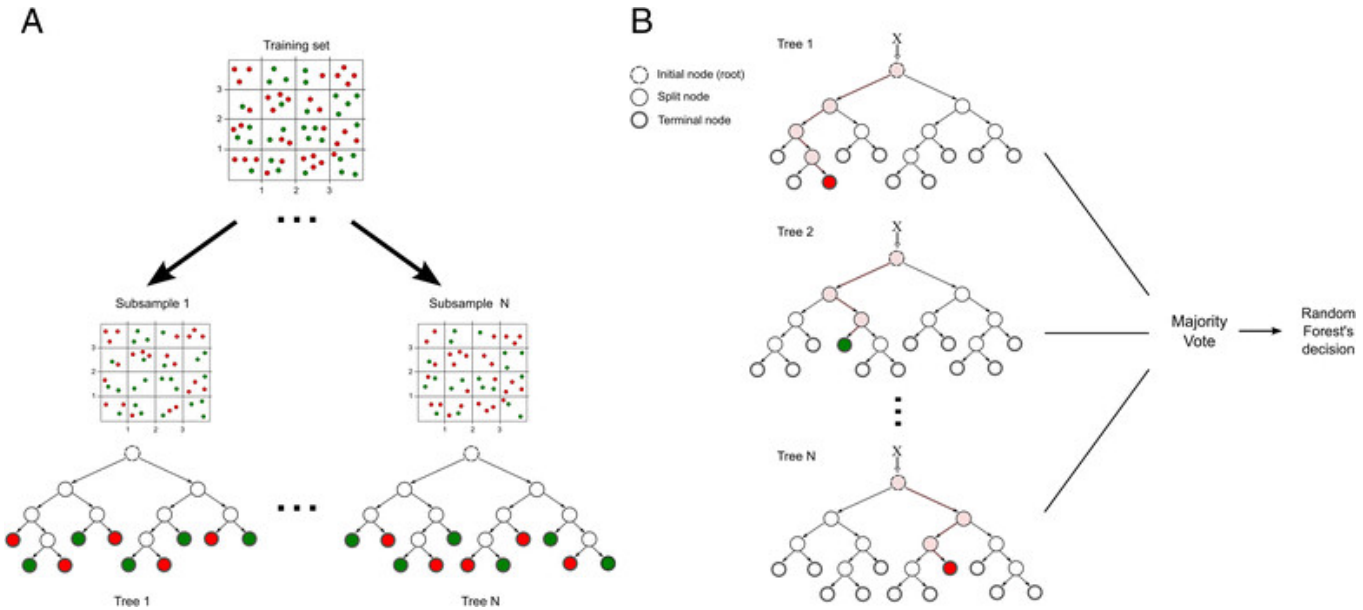
Random forests

Random Forest consists in generating multiple small decision trees from random subsets of the data (hence the name "Random Forest").
→ Each of the decision tree gives a biased classifier (as it only considers a subset of the data).
→ Each decision tree capture different trends in the data. This ensemble of trees is like a team of experts each with a little knowledge over the overall subject but thorough in their area of expertise. →

- In case of classification the majority vote is considered to classify a class.
- In case of Regression, we can use the avg. of all trees as our prediction.
- In addition to this, we can also weight some more decisive trees high relative to others by testing on the validation data.

Random forests -cont'd

Illustration



Extra trees ("extremely randomized trees")

Randomize both the learning subset, subset of features and thresholds :
→ : trades more bias for lower variance
→ : makes Extra-Trees much faster to train than regular Random Forests

Random forests -cont'd

Feature importance

Random Forests make it easy to measure the relative importance of each feature : Feature's importance is measured by looking at how much the tree nodes that use that feature reduce impurity on average (across all trees in the forest). More precisely, it is a weighted average, where each node's weight is equal to the number of training samples that are associated with it.

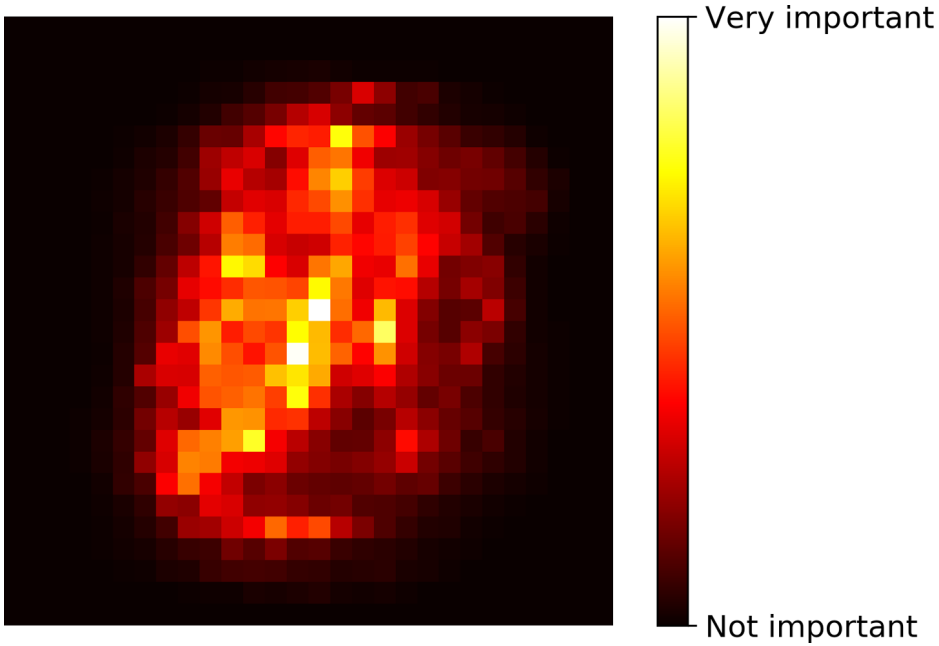
Remarks

• Feature importance may be also assessed (nor any kind of learner) by shuffling the values of a given feature, while keeping the others the same ('surrogate methods') : this is assumed to set the 'predictive importance' of the shuffled feature to almost zero. The difference in performance measured between shuffled / un-shuffled, quantifies the importance of the feature. But : This requires new test samples (out-of-bag) and is computationally expensive.
• Summing weighted Impurity gain at each node (involving a single feature) while computing the random forest involves only marginal computational increase. A disadvantage is that splits are biased towards variables with many classes, which also biases the importance measure.

feature importance example

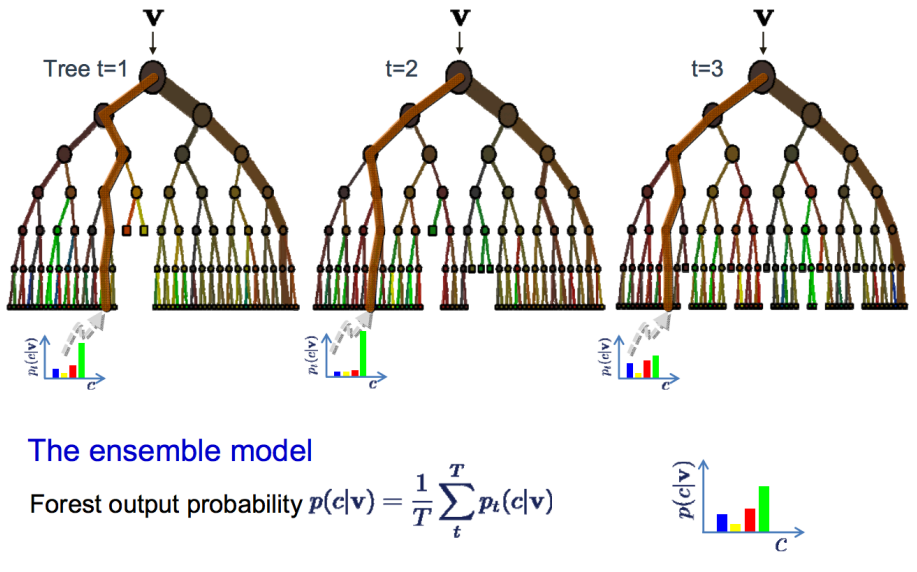
Handwritten characters,
16×16 pixels images

Pixel importance, according to
RF classifier with Gini Impurity



Random forest summary

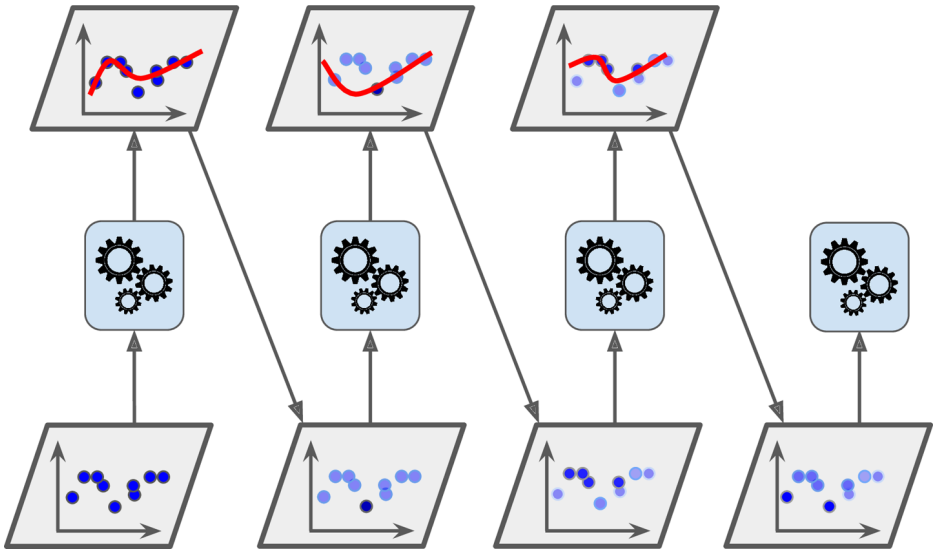
+ One of the most accurate decision models.
+ Can be used to extract variable importance.
+ Do not require feature engineering (scaling and normalization)
+ Capable of generating complex decision boundaries
− Risk of overfitting in case of noisy data.
− Unlike decision trees, results may be quite difficult to interpret.
− Hyperparameters needs good tuning for high accuracy :
    - Ntree : Nb of trees to grow in the forest.
    - Nfeatures : Nb of variables randomly sampled as candidates for each split for a given tree.
    - Replacement : Sampling done with or without replacement.



Sequential ensemble learning : Boosting

General idea : train predictors sequentially, each trying to correct its predecessor.

- A single training (sub)set is used to derive the predictor
- Important drawback to this sequential learning technique : it cannot be parallelized
- It does not scale as well as bagging or pasting (it uses a single large learning set)



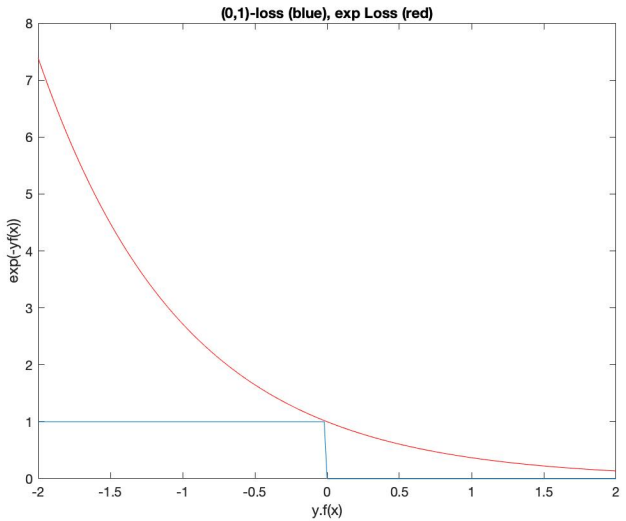
Boosting

The presentation below is mostly focused on a binary classification problem

Loss function

$$\begin{aligned} \{-1,1\} \times \mathcal{X} &\rightarrow \mathbb{R} \\ y, f(x) &\mapsto L(y, f(x)) \end{aligned}$$

In the following, emphasis is on $L(y, f(x)) = \mathrm{e}^{-yf(x)}$, for $y \in \{0, 1\}$.



Why exponential loss ?

- Convex upper bound differentiable approximation of $(0, 1)$-Loss
- Will lead to computationaly simple derivations (see below)

Boosting principle

Aim to find a greedy algorithm for fitting adaptive basis model of the form

$$F(x) = w_0 + \sum_{m=1}^{M} w_m f_m(x)$$

where the $f_m(.)$ are *weak* or *base* learners, $\{w_i\}$ are scalar weights.

Forward stagewise additive modeling

we want

$$f^*(x) = \arg\min_f \sum_{i=1}^{N} L(y_i, f(x_i))$$

where $L(y, f(x))$is the Loss function.

rk1 : if $L(y_i, f(x_i)$ is the quadratic loss, then

$$f^*(x) = \arg\min_f \mathbb{E}_{Y|x}[(Y - f(x))^2] = \mathbb{E}[Y|x]$$

$\Rightarrow$ requires to know $p(Y|x)$!

Boosting principle, -cont'd-

rk2 : For binary classification $L(y_i, f(x_i)$ is chosen to be the $[0,1]$-loss : it is not differentiable : it may be replaced by the convex upper bound given by the exponential loss, defined for $y \in \{-1, 1\}$ $L(y_i, f(x_i)) = \mathrm{e}^{-y_i f(x_i)}$.

Then

$$\begin{aligned} \frac{\partial}{\partial f}\mathbb{E}[\mathrm{e}^{-yf(x)}] &= \frac{\partial}{\partial f}\left[p(y=1|x)\mathrm{e}^{-f(x)} + p(y=-1|x)\mathrm{e}^{+f(x)}\right] \\ &= p(y=1|x)\mathrm{e}^{-f(x)} + p(y=-1|x)\mathrm{e}^{+f(x)} = 0 \end{aligned}$$

$\Rightarrow \mathrm{e}^{2f(x)} = \frac{p(y=1|x)}{p(y=-1|x)}$ then

$$f^*(x) = \frac{1}{2}\log\frac{p(y=1|x)}{p(y=-1|x)}$$

Boosting principle, -cont'd-

Sequential approach

Finding $f(.)$ is hard ; Boosting consists in solving it sequentialy :

Init :

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \Phi(x_i, \gamma))$$

where $\Phi(x, \gamma)$ is a (weak) learner, with (hyper)parameters γ.

Iteration m, $m \leq M$:

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta\Phi(x_i, \gamma))$$

where $f_m(x) \stackrel{def}{=} f_{m-1}(x_i) + \beta\Phi(x_i, \gamma)$

rk1 : Set M by monitoring performances
- on separate validation set
- by using BIC, AIC
- by recording wrt m Cost-complexity

rk2 : In practice, better perf. with shrinkage ν : $f_m(x) \stackrel{def}{=} f_{m-1}(x_i) + \nu\beta\Phi(x_i, \gamma)$, $0 < \nu \leq 1$.

$\Rightarrow$ The solution depends on the loss function.

Example : binary classification with exponential loss : AdaBoost(Shapire 1997)

$$y_i \in \{-1, +1\},\ L(y, f(x)) - = \mathrm{e}^{-yf(x)}$$

$$L_m(\Phi) \stackrel{def}{=} \sum_{i=1}^{N} \mathrm{e}^{-y_i[f_{m-1}(x_i)+\beta\Phi(x_i)]} = \sum_{i=1}^{N} w_i^{(m)} \mathrm{e}^{-y_i\beta\Phi(x_i)}$$

with $w_i^{(m)} = \mathrm{e}^{-y_i f_{m-1}(x_i)}$, then

$$\begin{aligned} L_m(\Phi) &= \mathrm{e}^{-\beta} \sum_{y_i=\phi(x_i)} w_i^{(m)} + \mathrm{e}^{+\beta} \sum_{y_i\neq\phi(x_i)} w_i^{(m)} \\ &= (\mathrm{e}^{+\beta} - \mathrm{e}^{-\beta}) \sum_i w_i^{(m)} \mathbb{I}(y_i \neq \Phi(x_i)) + \mathrm{e}^{-\beta} \sum_i w_i^{(m)} \end{aligned}$$

$\Rightarrow$ choose $\Phi_m = \arg\min_\Phi w_i^{(m)} \mathbb{I}(y_i \neq \Phi(x_i))$
$\Phi_m()$ can be mearned by a weak learner, assuming that it can integrate the weights $w_i^{(m)}$. For decision trees, $w_i^{(m)}$ are integrated in the evaluation of the impurity index.

AdaBoost -cont'd

Inserting $\Phi_m()$ in L_m and setting the derivative wrt β to zero,

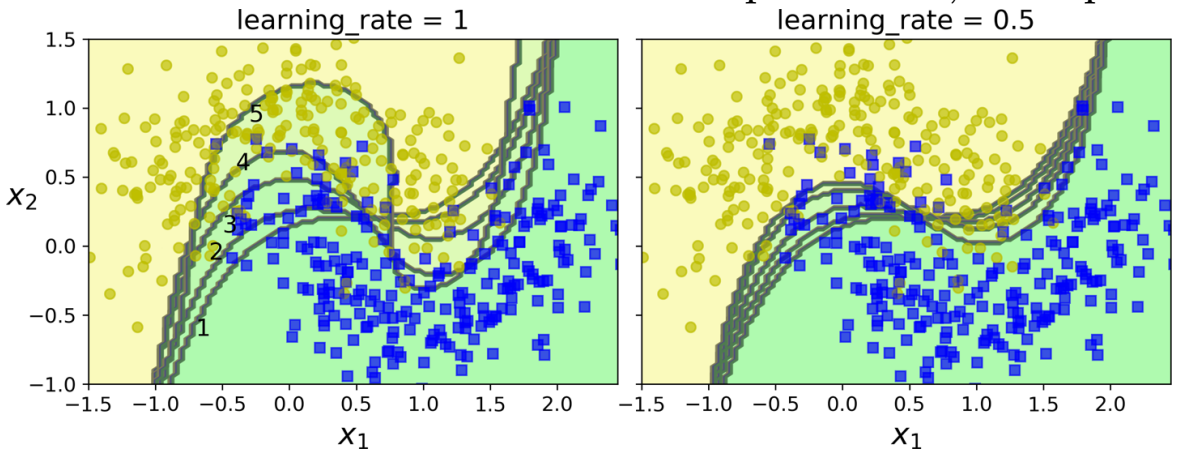
$$\frac{\partial}{\partial \beta}[(\mathrm{e}^{+\beta} - \mathrm{e}^{-\beta}) \sum_i w_i^{(m)} \mathbb{I}(y_i \neq \Phi(x_i)) + \mathrm{e}^{-\beta} \sum_i w_i^{(m)}] = 0$$

gives : $\beta = \frac{1}{2} \log \frac{1-\varepsilon_m}{\varepsilon_m}$ where $\varepsilon_m = \frac{\sum_i w_i^{(m)} \mathbb{I}(y_i \neq \Phi(x_i))}{\sum_i w_i^{(m)}}$

AdaBoost algorithm

- **Initialization :** $w_i^{(0)} = \frac{1}{N}, \forall i \in \{1, \ldots, N\}$
- **Iterate :** for $m = 1, \ldots, M$ do
  - fit $\Phi_m(x)$ to training set with weights $w_i^{(m)}$
  - compute $\varepsilon_m = \frac{\sum_i w_i^{(m)} \mathbb{I}(y_i \neq \Phi(x_i))}{\sum_i w_i^{(m)}}$
  - compute $\beta_m = \frac{1}{2} \log \frac{1-\varepsilon_m}{\varepsilon_m}$
  - $w_i^{(m+1)} \leftarrow w_i^{(m)} \exp\left(\beta_m \mathbb{I}(y_i \neq \Phi(x_i))\right)$
- **return :** $f(x) = \mathrm{sign}[\sum_{m=1}^{M} \beta_m \Phi_m(x)]$

Decision boundaries of consecutive predictors, example :



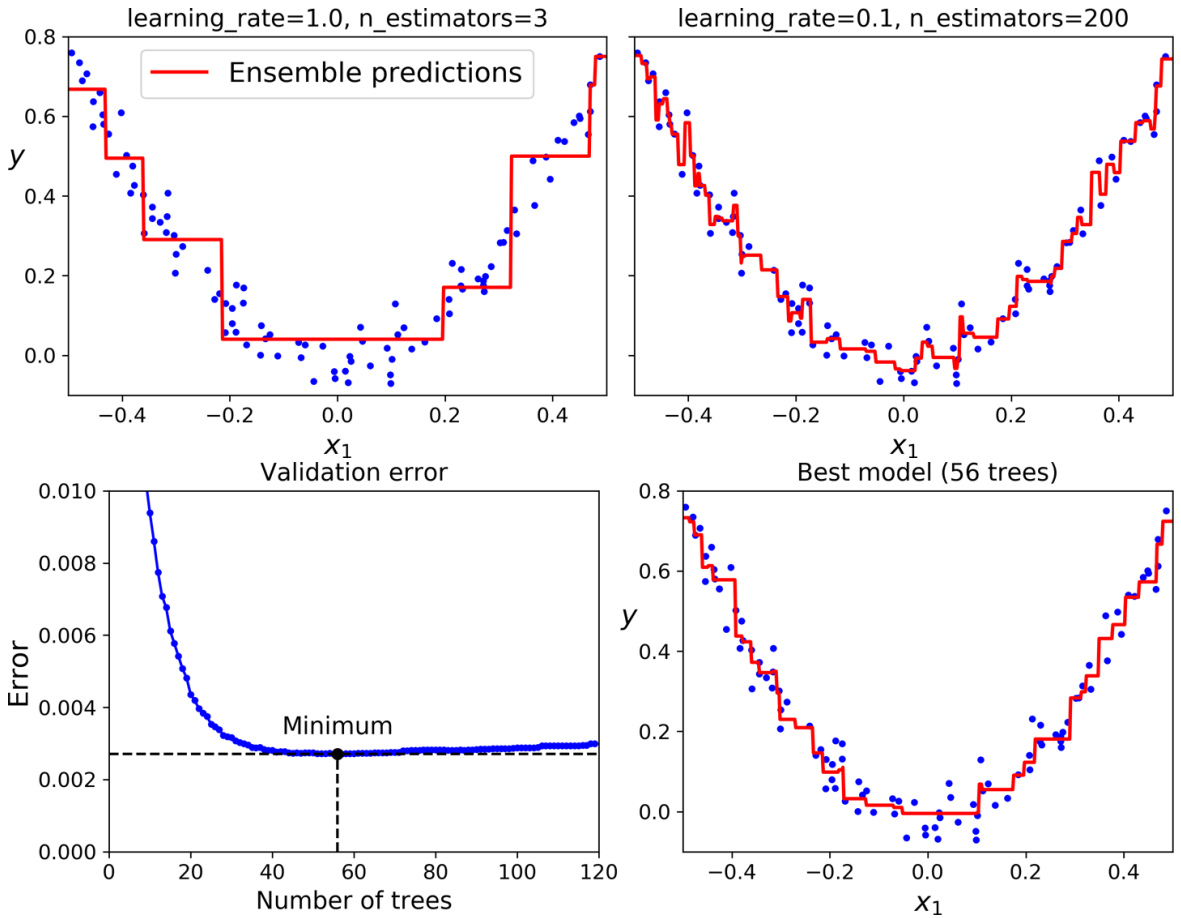
Gradient boosting

Generic version, sequentially adding predictors to an ensemble, each one correcting its predecessor. Instead of adapting instance weights, tries to fit the new predictor to the residual errors made by the previous predictor

Gradient boosting algorithm

- **Initialization :** $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \Phi(x_i, \gamma)$
- **Iterate :** for $m = 1, \ldots, M$ do
  - compute the gradient residual using $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$
  - use the weak learner to compute $\gamma_m = \arg\min_\gamma \sum_{i=1}^N (r_{im} - \Phi(x_i; \gamma_m))^2$
  - update $f_m(x) = f_{m-1}(x) + \nu\Phi(x, \gamma_m)$
- **return :** $f(x) = f_M(x)$

Ensemble learning : setting M ?



Scikit-learn usefull codes for this section :

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
```