Ensemble learning from theory to practice

Lecture 2: Bagging and Random Forests

Myriam Tami

myriam.tami@centralesupelec.fr

January - March, 2022

CentraleSupélec

Reminder

In the first course, we saw

- **Goal of ensemble learning methods**:
  combine multiple weak learners in order to improve robustness and prediction performances
- Decision tree is an example of weak learners

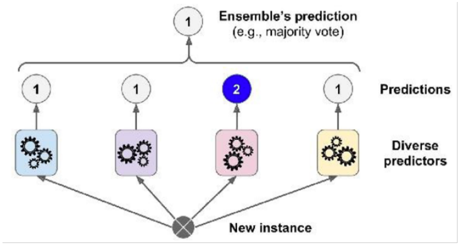


Figure: source: Hands-On Machine Learning with Scikit-Learn and TensorFlow, A. Géron

Why to combine? An intuition

Binary classification $Y \in \{-1, 1\}$, input variables $X \in \mathbb{R}^d$
We have a set of K independent initial classification methods $(f_b)_{1 \leq b \leq B}$ such as $\forall b$,

$$P\left\{f_b(X) \neq Y\right\} = \varepsilon$$

By aggregating these methods and predicting

$$F(X) = sign\left(\sum_{b=1}^{B} f_b(X)\right)$$

By the Hoeffding inequality (not on the course program), the probability of error of F is:

$$P\left\{F(X) \neq Y\right\} \leqslant exp\left(-\frac{1}{2}B\left(2\varepsilon - 1\right)^2\right)$$

which tends towards zero exponentially in B

Why to combine? Another intuition

Assume that $X_1, \ldots, X_B$ are B iid random variables of mean $\mu = E[X_1]$ and of variance $\sigma^2 = V[X_1] = E[(X_1 - \mu)^2]$

Consider the new variable/estimator (empirical mean)

$$\bar{X} := \frac{1}{B} \sum_{b=1}^{B} X_b$$

- The expectation does not change, $E[\bar{X}] = \mu$ (no bias i.e., $E[\bar{X} - \mu] = 0$, the expected value of the estimator matches that of the parameter (no error))
- The variance is reduced **thanks to the decorrelation of the random variables**, $V[\bar{X}] = \frac{1}{B}\sigma^2$

This is of interest of decision trees: we have seen (cf. Course 1) that large and unpruned trees have a small bias but a large variance

Bagging (Bootstrap AGGregatING)

Introduced by Breiman [Breiman, 1996]

Based on two key points : **boostrap** and **aggregation**

- We know that the aggregation of independent initial predictive methods (base learners) leads to a significant reduction in error of prediction and variance
  $\Rightarrow$ Get initial methods as independent as possible
- Naive idea: train our "base learners" (ex: CART) on subsets of **disjoint** observations of the training set
- Problem: the training set is not infinite $\rightarrow$ the "base learners" will have too little data and poor performance

That is where bootstrapping is useful

Bagging idea

Bagging create training subsets using bootstrap sampling [Tibshirani and Efron, 1993]

Boostrapping

To create a new “base learner" f_b,

1. we randomly draw with replacement a dataset $\mathcal{D}_b$ of n_{train} observations from the training set
2. we learn the method (ex: CART) on it
   $\rightarrow$ the “base learner" f_b is obtained

Note: each $\mathcal{D}_b$ has the same size as the original training set

Bagging idea

Bagging

Consists to

1. Do boostrapping B times producing
   - B bootstrap datasets $\mathcal{D}_b$
   - then B predictors (“base learners") f_b for each of these datasets
2. Aggregate the predictors
   - In the regression case (average),

$$f_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x) \tag{1}$$

   - In the classification case (majority vote over trees),

$$f_{bag}(x) = \underset{1\leqslant c\leqslant C}{\operatorname{argmax}} \left(\frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{f_b(x)=c\}} \right) \tag{2}$$

Bagging diagram

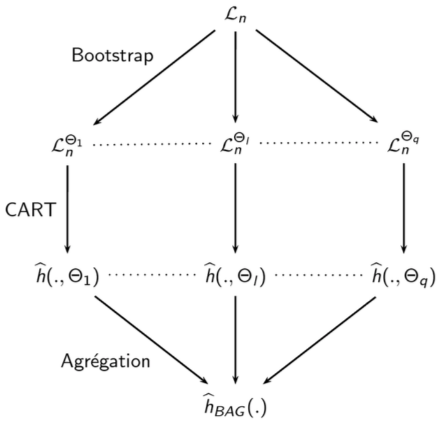

$\mathcal{L}_n$
Bootstrap
$\mathcal{L}_n^{\Theta_1}$
$\mathcal{L}_n^{\Theta_l}$
$\mathcal{L}_n^{\Theta_q}$
CART
$\widehat{h}(., \Theta_1)$
$\widehat{h}(., \Theta_l)$
$\widehat{h}(., \Theta_q)$
Agrégation
$\widehat{h}_{BAG}(.)$

Figure: Illustration of the bagging principle (with $\mathcal{L}_n^{\Theta_l} = \mathcal{D}_b$, $\hat{h}(., \Theta_b) = \hat{f}_b$ and $\hat{h}_{bag} = \hat{f}_{bag}$)

Random forests

- Method introduced by Leo Breiman in 2001 [Breiman, 2001]
- Based on older ideas: Bagging [Breiman, 1996], decision trees CART [Breiman et al., 1984]
- Proofs of the convergence [Biau et al., 2008]
- Random forests method belongs to the family of ensemble methods

Random forests (notations)

- $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ the learning set, each (x_i, y_i) is independent, realization from the random variables (X, Y)
- $X \in \mathbb{R}^d$ the input variables; $Y \in \mathcal{Y}$ the output variable, $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = \{1, \ldots, C\}$ for classification

Goal: build a predictor $\hat{f} : \mathbb{R}^d \to \mathcal{Y}$

Random forests idea

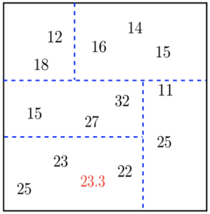
$\left\{\hat{f}_b(., \Theta_b), 1 \leqslant b \leqslant B\right\}$ set of decision tree predictors,
$(\Theta_b)_{1 \leqslant b \leqslant B}$ characterises the *b*th tree in terms of split variables, cutpoints at each node and terminal-node value,
Random forests predictor $\hat{f}$ obtained by aggregating the set of trees

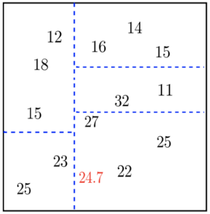
- for regression,
$$\hat{f}(x) := \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x, \Theta_b) \tag{3}$$
- for classification,
$$\hat{f}(x) := \underset{1 \leqslant c \leqslant C}{\operatorname{argmax}} \left(\frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{\hat{f}_b(x, \Theta_b) = c\}} \right) \tag{4}$$

Random forests

- Random forests consist of growing a large number (ex: 400) of randomly constructed decision trees before aggregating them
- In statistical terms, if the trees are decorrelated, this reduces the variance of the predictions

Naïve example with 2 trees,





Averaging these regression trees allows the prediction $\frac{24.7+23.3}{2} = 24$

source: *Arbres de décision et forêts aléatoires*, P. Gaillard, 2014

The problem of correlation between trees

- Bagging idea: aggregate many noisy but approximately unbiased[1] tree models to reduce the variance
- However, there is necessarily some overlap between bootstrapped datasets
  $\Rightarrow$ the trees corresponding to each of them are correlated
- Intuition: if B trees $f_b(x)$ are identically distributed, of variance σ^2, with a correlation coefficient $\rho = Corr(f_b(x), f_{b'}(x)), \ \forall b \neq b'$, the variance of their mean is then,

$$V\left(f_{bag}\left(X\right)\right) = \rho\sigma^2 + \frac{\left(1-\rho\right)\sigma^2}{B} \tag{5}$$

Thus, the variance cannot be shrink below $\rho\sigma^2$
$\Rightarrow$ Disadvantage of bagging

[1] if sufficiently deep

Create low correlated trees

We saw the disadvantage of

- Bootstrapping: rather than using all the data to build the trees, we choose randomly for each tree a subset (with possible repetition) of the training data.

Let introduce the improvement proposed by random forests: is to lower the correlation between trees (without increasing the variance too much) using an additional step of randomization,

- a random choice of the input feature j used to split each node during the tree-growing process

Definition of Random forests

Definition (Random forests)

A random forest is a set of trees growing on a boostrapped learning data set and where, before each split, a set of $m \ll d$ input variables (or features) is randomly selected as candidates for splitting

Note: m is the same for all the nodes of all the trees of the forest but, of course, the variables considered in each node for the choice of the best split changes randomly

Algorithm of Random forests

```
1: Require: A dataset $\mathcal{D} = \left\{(x_i, y_i)_{1 \leqslant i \leqslant n}\right\}$, the size $B$ of the ensemble, the number
   $m$ of candidates (features) for splitting
2: for $b = 1$ to $B$ do
3:     Draw a bootstrap dataset $\mathcal{D}_b$ of size $n$ from the original training set $\mathcal{D}$
4:     Grow a random tree $\hat{f}_b$ using the bootstrapped dataset:
5:     repeat
6:     for all terminal node do
7:         Select $m$ variables among $d$, at random
8:         Pick the best variable and split-point couple among the $m$
9:         Split the node into two child nodes
10:    end for
11:    until the stopping criterion is met (e.g., minimum number of sample per node
       reached)
12: end for
   return: the ensemble of $B$ trees
```

Algorithm 1: Pseudo-code to build Random forests for regression or classification

Random forests in practice

- Intuitively, reducing m will reduce the correlation between any pairs of trees in the ensemble
  $\Rightarrow$ reduce the variance of the average (cf. Eq. (5))
- However, the corresponding hypothesis space will be smaller, leading to an increased bias

Heuristics,

- for regression, choose $m = \lfloor \frac{d}{3} \rfloor$ and a minimum node size of 5
- for classification, choose $m = \lfloor \sqrt{d} \rfloor$ and a minimum node size of 1

For further information about random forests, you can refer to [Hastie et al., 2009] (Chap. 15) that provides a bias-variance analysis

OOB-error (Out-Of-Bag error)

OOB error (out of bootstrap samples) - the principle

To predict y_i, we only aggregate the predictors $\hat{f}_b(., \Theta_b)$ built on bootstrap samples not containing (x_i, y_i)

- for regression,

$$\text{OOB-error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  where $\hat{y}_i$ from an aggregation of $\hat{f}_b$ built on $\mathcal{D}_b \setminus (x_i, y_i)$

- for classification,

$$\text{OOB-error} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{y_i \neq \hat{y}_i}$$

OOB-error

Many advantages,

- An OOB-error estimate is almost identical to that obtained by N-fold cross-validation
- Random forests can be fit in one sequence, with cross-validation being performed along the way
- Once the OOB-error stabilizes, the training can be terminated and *B* value is obtained/ tunned

Variable Importance (VI)

- Random Forests (RF) allow to rank the explanatory variables in order of importance for the prediction
- In the RF framework, **permutation importance indices** are preferred to **total decrease of node impurity measures** already introduced in Breimanet al.(1984)
- The default `Scikit-learn`'s feature importances is based on the decrease of node impurity

Variable Importance based on impurity decrease

For one tree,

- Variable Importance (VI) of X_j is calculated by the sum of the decrease in error/impurity when split by the variable X_j

  e.g., if X_j is used 2 times to split a terminal node in the tree $\rightarrow$ you will sum these two decreases in Gini index (or cross-entropy, etc) to obtain its VI

- The relative importance is the VI divided by the highest VI value (normalization)
  $\Rightarrow$ Values are bounded between 0 and 1

In the case of RF,

- we are talking about **averaging** the decrease in impurity over trees

Variable Importance based on impurity decrease

Pros,

- Fast calculation
- Easy to obtain via `Scikit-learn`: one command `feature_importances_`

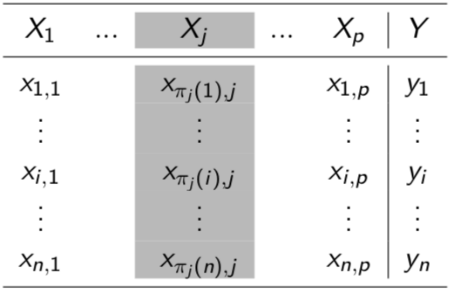
Cons,

- Biased approach: it has a tendency to inflate the importance of continuous features or high-cardinality categorical variables

Variable Importance based on permutation

This approach directly measures the feature importance of an input variable X_j by observing how random permutation of its values (thus preserving the distribution of the variable) influences model performance

Goal: measure the prediction strength of each variable



X_1	$\dots$	X_j	$\dots$	X_p	Y
$x_{1,1}$		$x_{\pi_j(1),j}$		$x_{1,p}$	y_1
$\vdots$		$\vdots$		$\vdots$	$\vdots$
$x_{i,1}$		$x_{\pi_j(i),j}$		$x_{i,p}$	y_i
$\vdots$		$\vdots$		$\vdots$	$\vdots$
$x_{n,1}$		$x_{\pi_j(n),j}$		$x_{n,p}$	y_n

Figure: Illustration of the values permutation from one variable (source: *Arbres CART et Forêts aléatoires - Importance et sélection de variables*, R. Genuer and J-M. Poggi, 2016)

Variable Importance based on permutation

The process is the following,

1. Grow the RF on the learning set
2. Record the OOB-error E
3. Permute at random the j-th variable values of these data
4. Pass this modified dataset to the RF again to obtain predictions
5. Compute the OOB-error on this modified dataset
6. The VI of X_j is the difference between the benchmark score E and the one from the modified (permuted) dataset

$\Rightarrow$ The more the increase of OOB error is, the more important is the variable

Variable Importance based on permutation

Pros,

- Reasonably efficient
- Reliable technique
- No need to re-train the model at each modification of the dataset

Cons,

- More computationally expensive than the default `feature_importances_`
- Permutation importance overestimates the importance of correlated predictors [Strobl et al., 2008]

Anomalies detection

- RFs are well suited to detecting outliers [Liu et al., 2008]
- These are indeed quickly isolated in a separate leaf
- The anomaly score of an observation x_i is determined approximately by the average length of the path from x_i to the leaves of trees in the forest
- The shorter the path, the more likely the observation is atypical

Pros and cons of Random forests

Pros

- no over-learning
- usually: better performance than decision trees
- direct computation of the"Out-of-Bag" error: cross validation not required
- hyper-parameters (B, m) easy to tune

Cons

- black box: difficult to interpret
- slower training

Extremely randomized trees

Randomization can be pushed further with extremely randomized forests

- Method introduced by [Geurts et al., 2006]
- It is a RF with two differences,
  - $m < d$ of the input variables are selected at random and for each of these variables **a split point is chosen at random**
  - The full learning set $\mathcal{D}$ is used to growth each tree (instead of a bootstrapped learning set $(\mathcal{D}_b)_{1 \leq b \leq B}$)

Impact on correlation and bias

Theses two differences:

1. Using the full learning set
   $\Rightarrow$ achieve a lower bias
   But, the price is an increased variance

   That should be compensated by the randomization of split-points,
2. choosing also the split-point at random
   $\Rightarrow$ reduce the correlation between trees to reduce the variance of the average of the ensemble more strongly

Algorithm of Extremely Randomized Forest

```
1:  Require: A dataset $\mathcal{D} = \left\{(x_i, y_i)_{1 \leqslant i \leqslant n}\right\}$, the size $B$ of the ensemble, the number
    $m$ of candidates for splitting
2:  for $b = 1$ to $B$ do
3:      Grow a random tree using the original dataset $\mathcal{D}$:
4:      repeat
5:      for all terminal node do
6:          Select $m$ variables among $d$, at random
7:          for all sampled variables do
8:              Select a split at random
9:          end for
10:         Pick the best variable and split-point couple among the $m$ candidates
11:         Split the node into two child nodes
12:     end for
13:     until the stopping criterion is met (e.g., minimum number of sample per node
        reached)
14: end for
    return: the ensemble of $B$ trees
```

Algorithm 2: Pseudo-code describing Extremely Randomized Forest approach

Extremely Randomized Forest

Advantages of this approach,

- Empirically, it often provides better results than RFs
- Lower computational complexity compared to RFs (one chooses the split among the m randomly drawn split-points)

Disadvantages

- Black box: difficult to interpret

References I

Biau, G., Devroye, L., and Lugosi, G. (2008).
Consistency of random forests and other averaging classifiers.
Journal of Machine Learning Research, 9(Sep):2015–2033.

Breiman, L. (1996).
Bagging predictors.
Machine learning, 24(2):123–140.

Breiman, L. (2001).
Random forests.
Machine learning, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984).
Classification and regression trees.
CRC press.

Geurts, P., Ernst, D., and Wehenkel, L. (2006).
Extremely randomized trees.
Machine learning, 63(1):3–42.

Hastie, T., Tibshirani, R., and Friedman, J. (2009).
The elements of statistical learning: data mining, inference, and prediction.
Springer Science & Business Media.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008).
Isolation forest.
In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.

References II

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008).
Conditional variable importance for random forests.
BMC bioinformatics, 9(1):307.

Tibshirani, R. J. and Efron, B. (1993).
An introduction to the bootstrap.
Monographs on statistics and applied probability, 57:1–436.