Machine Learning
Trees, Bagging and Random forests

Audrey Poterie
Univ. Bretagne Sud – LMBA Vannes .

October 19, 2021

Part II : Model averaging - Bagging and random forests

Content

Introduction: model averaging

Bagging

Random forests

- Algorithm
- Out-Of-Bag sample
- Variable importance

Model averaging

Motivation: Decision trees can be simple, but often produce unstable models with weak performance compared to other supervised methods.
➜ Use decision trees as building blocks to construct more powerful prediction models ➜ Model averaging.

Model averaging

Motivation: Decision trees can be simple, but often produce unstable models with weak performance compared to other supervised methods.
➜ Use decision trees as building blocks to construct more powerful prediction models ➜ Model averaging.

Model averaging

- **Bagging** [Breiman, 1996]: fit many large trees to bootstrap-resampled versions of the train set, and average the predictions.
- **Random forests** [Breiman, 2001]: a fancier version of bagging that uses features sampling.
- **Boosting** [Freund et al., 1996]: fit many large or small trees to reweighted versions of the train set (no bootstrap resampling) and use a weighted average of the predictions.

Model averaging

Some remarks:

- In general (in terms of performance), Boosting $\succ$ Random forests $\succ$ Bagging $\succ$ single decision tree.
- Synonym of model averaging: ensemble learning.
- Here, we will only focus on bagging and random forests.

Content

Introduction: model averaging

Bagging

Random forests

- Algorithm
- Out-Of-Bag sample
- Variable importance

What is bagging ?

Principle: Bagging or **B**ootstrap **Agg**regat**ing** averages a given procedure over many samples.
➜ *Example: build lots of large and unpruned CART trees to bootstrap resampled versions of a data set.*

Two steps: (1) bootstrappping, (2) aggregation.

Motivations:

- To improve model stability and so model accuracy.
- To avoid overfitting.
  ➜ ⚠ Important since we use large and unpruned trees[1].

[1] Reminder: very large trees = trees that fit almost perfectly the data, i.e. trees with almost no bias but large variance.

Bootstrapping

Data available: only one data set $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots (\mathbf{X}_n, Y_n)\}$ to train the model.

⚠ We need many samples and we cannot create and/or get additionnal data.

Solution: use bootstrapping = generate "new" data sets by randomly drawing observations (with or without replacement) into the original data set $\mathcal{D}_n$.
➜ We say that we generate bootstrap samples.

Bootstrapping

Bootstrap algorithm

- **Input**: one data set $\mathcal{D}_n$ with 8 observations.

$\mathcal{D}_n$	obs 1	obs 2	obs 3	obs4	obs 5	obs 6	obs 7	obs 8

- **Do bootstrapping**: repeat independently the following procedure B times: draw randomly and with replacement observations into $\mathcal{D}_n$.

$\mathcal{D}_n^1$	obs2	obs 6	obs 5	obs 5	obs 2	obs 4	obs 8	obs 7
$\mathcal{D}_n^2$	obs 6	obs 8	obs 2	obs 4	obs 2	obs 7	obs 8	obs 1
				$\vdots$				
$\mathcal{D}_n^B$	obs 7	obs 8	obs 3	obs 7	obs 2	obs 3	obs 3	obs 1

- **Output**: B bootstrap-resampled versions of the original data set $\mathcal{D}_n \Leftrightarrow B$ "new" data sets made up of observations from $\mathcal{D}_n$:

$$\mathcal{D}_n^1, \ldots, \mathcal{D}_n^B.$$

Bootstrapping

Important: the B boostrap samples are drawn independently.

Two sampling strategies:

1. Draw with replacement $a_n = n$ observations into $\mathcal{D}_n$, ($n =$ size of $\mathcal{D}_n$).
2. Draw without replacement $a_n \leq n$ observations into $\mathcal{D}_n$ (also called subsampling).

Bagging algorithm

Reminder: Bagging = bootstrapping + aggregation.

Bagging algorithm

Inputs:

- An original sample $\mathcal{D}_n$.
- A learning method/algorithm: here CART algorithm.
- Bagging parameters: B, a_n.

For b = 1 to B, repeat INDEPENDENTLY the two following steps:

1. Draw a bootstrap sample $\mathcal{D}_n^b$ of size a_n.
2. Build a large and unpruned CART tree using $\mathcal{D}_n^b$ and call it T_n^b.

end.

Output: For an observation **x**, the bagging prediction = aggregation of the predictions obtained with the B trees.

Bagging algorithm

Bagging prediction

For an observation **x**, the bagging prediction is:

- *In regression:* the prediction average over the **B** trees

$$f_n^B(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f_n^b(\mathbf{x}),$$

  where $f_n^b(\mathbf{x})$ is the prediction of the bth tree T_n^b for observation **x**.

- *In classification:* the majority class over the **B** trees

$$f_n^B(\mathbf{x}) = \mathit{majority\ vote}\{\hat{C}_b(\mathbf{x})\}_1^B,$$

  where $\hat{C}_b(\mathbf{x})$ is the class prediction of the bth tree T_n^b for observation **x**.

Important remarks

- The B boostrap samples are independent and the B trees are built independently.

- Choose B as large as possible, often around 500.
  ➜ Larger B is, more stable bagging predictions are.

- Bagging does not impact the bias: suppose all trees have the same bias θ then the bias of the bagging model will still equal θ.

- Bagging reduces model variance and so model stability (bagging properties).

Variance reduction with bagging

Idea: suppose we have B regression trees with same variance σ^2 and a positive pairwise correlation ρ. Then the variance of the tree average (= variance of bagging for regression) is

$$V_{bagging} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

As B increases, the second term disappears and so we have

$$V_{bagging} \approx \rho\sigma^2.$$

Since $0 < \rho < 1$, we obtain

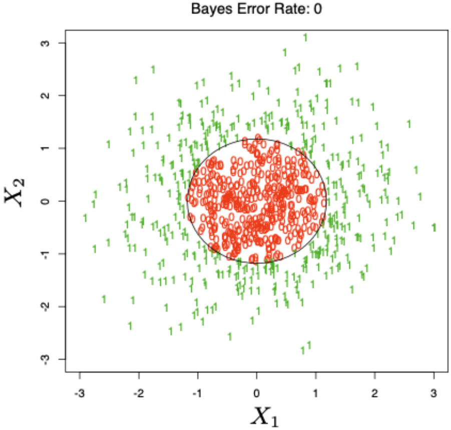
$$V_{bagging} < \sigma^2.$$

➜ **Conclusion:** the bagging variance is (generally) lower than the variance of a single tree:

$$V_{bagging} < V_{single\ tree}.$$

Illustration with a toy example (1/5)

- A simple deterministic problem.
- No noise: Bayes error=0.



➜ We will compare performance of a single CART tree and bagging.

Example drawn from https://web.stanford.edu/ hastie/TALKS/boost.pdf

Illustration with a toy example (2/5)

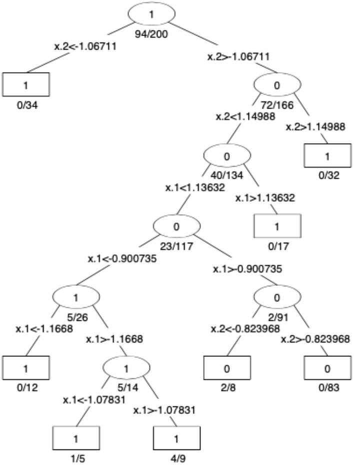


Figure 1: The single CART tree based on a train set of size 200.

Illustration: a toy example (3/5)

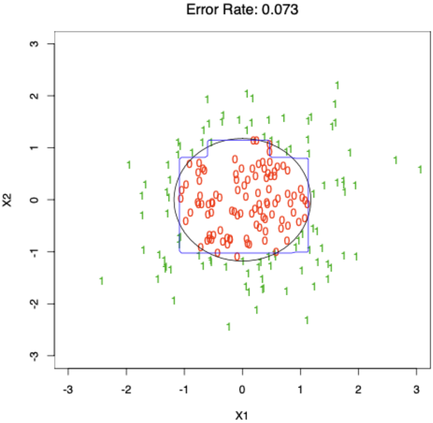
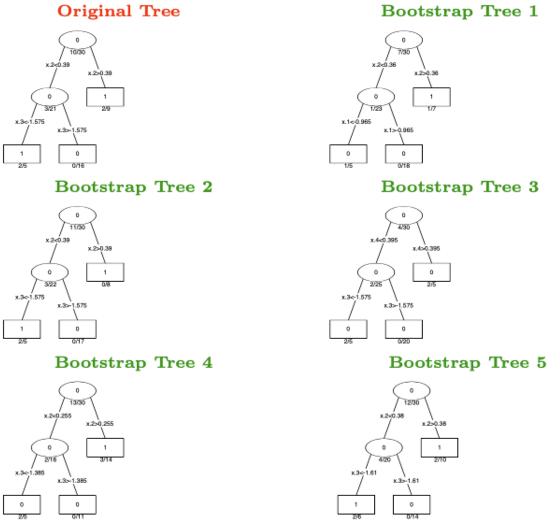


Figure 2: The partition associated with the previous single CART tree.

Illustration: a toy example (4/5)



Original Tree

Bootstrap Tree 1

Bootstrap Tree 2

Bootstrap Tree 3

Bootstrap Tree 4

Bootstrap Tree 5

Figure 3: Single tree vs. Bagging.

Illustration: a toy example (5/5)

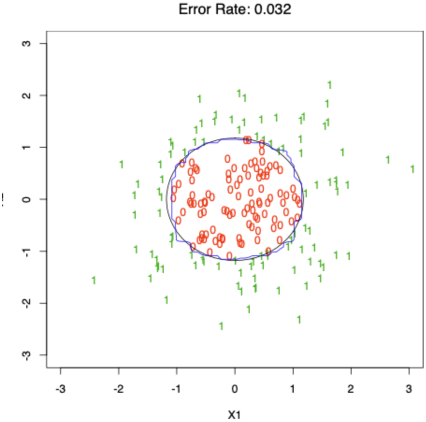


Figure 4: The partition obtained with bagging.

Illustration: a toy example (5/5)

- ➜ The decision boundaries are smoothers (consequence of aggregation).
- ➜ Bagging error is lower.

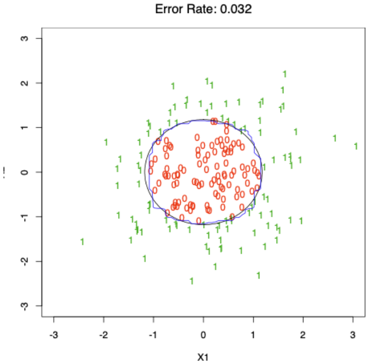


Figure 4: The partition obtained with bagging.

Content

Introduction: model averaging

Bagging

Random forests

- Algorithm
- Out-Of-Bag sample
- Variable importance

Content

Introduction: model averaging

Bagging

Random forests

- **Algorithm**
- Out-Of-Bag sample
- Variable importance

What is random forests ?

Author: Phill Cutler

Principle: Random forest = bagging + features sampling.

Motivations:

- Same as bagging: performance improvement (by reducing variance) and no overfitting.
- Additional motivation: greater variance reduction than bagging.
  ➜ **Strategy:** encouraging diversity among trees by using random features sampling.

Variance reduction with random forests

Idea: suppose we have B regression trees with same variance σ^2 and a positive pairwise correlation ρ. Then the variance of the tree average ($=$ variance of bagging for regression) is

$$V_{bagging} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

As B increases, the 2nd term disappears ($=$ *gagging benefit*) and so we have

$$V_{bagging} \approx \rho\sigma^2.$$

➜ **Random forests idea:** use features sampling to reduce the tree correlation ρ:

$$\begin{aligned} \rho_{rf}\sigma^2 &< \rho\sigma^2 \\ V_{rf} &< V_{bagging} \end{aligned}$$

with $\rho_{rf} < \rho$ denoting the tree correlation after using features sampling.

➜ **Conclusion:** Random forests achieves (generally) greater variance reduction than bagging.

Features sampling

Features sampling in random forests

- Modification of the CARTalgorithm.
- Features sampling performed during the tree building process and specifically when splitting a node:
  1. Random selection of `max_features` $< d$ features among the d features.
  2. Selection of the best split by using only this subset.

Remarks: we can say that RF use rCART trees with a 'r' for random.

Random forests algorithm

Reminder: Random forests = Bagging + Features sampling.

RF algorithm

Inputs:

- An original sample $\mathcal{D}_n$.
- A learning method/algorithm: CART algorithm.
- RF parameters: B, a_n, `max_features`, `nodesize` .

For b = 1 to B, repeat INDEPENDENTLY the two following steps:

1. Draw a bootstrap sample $\mathcal{D}_n^b$ of size a_n.
2. Build a large and unpruned rCART tree T_n^b using $\mathcal{D}_n^b$ by repeating the following steps on each terminal node until reaching the minimun node size (`nodesize`):
   To split a node,
   a) Draw randomly `max_features` features.
   b) Select the best split based only on the `max_features` selected features.

end.

Output: For an observation **x**, the RF prediction = aggregation of the predictions obtained with the B trees.

Random forests algorithm

RF prediction

For an observation $\mathbf{x}$, the RF prediction is:

- *In regression:* the prediction average over the B trees

$$f_n^{RF}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f_n^b(\mathbf{x}),$$

where $f_n^b(\mathbf{x})$ is the prediction of the bth tree T_n^b for observation $\mathbf{x}$.

- *In classification:* the majority class over the B trees

$$f_n^{RF}(\mathbf{x}) = majority\ vote\{\hat{C}_b(\mathbf{x})\}_1^B,$$

where $\hat{C}_b(\mathbf{x})$ is the class prediction of the bth tree T_n^b for observation $\mathbf{x}$.

Parameters in random forests

- B = the number of trees in the forest.
  ➜ Take B large (default value: $B = 500$) to obtain stable/robust estimate.

- $a_n \leq n$ = the size of each bootstrap sample and the boostrap strategie.
  ➜ Two usual bagging strategies:
  (1) Choose $a_n = n$ and draw with replacement,
  (2) Choose $a_n \leq n$ and draw without replacement.

- `nodesize` = the minimum number of observations required to split a node
  .
  ➜ Take `nodesize` small to obtain trees with small bias and large variance.

- `max_features` = the number of features randomly selected when splitting a node.
  ➜ Almost the most important parameter to be tuned.

What value for `max_features` ?

Parameter `max_features`: "*slighlty controls*" the bias-variance tradeoff for the forest.

Explanation:

- Smaller `max_features`: more different and less correlated trees (smaller $|\rho|$) but tree that do not fit well the data (larger bias).
- Larger `max_features`: trees that fit correctly the data (= smaller bias) but tree more similar and so more correlated (larger $|\rho|$).

Default values: `max_features`$= d/3$ in regression and `max_features`$= \sqrt{d}$ in classification.

➜ Often reported to be good choices.

➜ Yet, optimal value for `max_features` depends mainly on the data: use of the out-of-bag samples to tune this parameter.

Content

Introduction: model averaging

Bagging

Random forests

- Algorithm
- **Out-Of-Bag sample**
- Variable importance

Random forest performance

As for other machine learning algorithms, we need a criterion to measure performances of a random forest.

Common criteria (reminder):

- The quadratic loss for regression: $\frac{1}{n}\sum_{i=1}^{n}(y_i - f_n^{RF}(\mathbf{x}))^2$.
- The misclassification error for classification: $\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{y_i \neq f_n^{RF}(\mathbf{x})}$.

Reminder: to be a good proxy of the *true* error, these criteria must be evaluated on a test set or by using cross validation, leave-one-out, etc.

Boostrapping-step asset: performance criteria can be evaluated directly on the original dataset $\mathcal{D}_n$ by using the Out of Bag samples (OOB).
➜ With RF: no need for a test set or to use strategies such as cross validation or leave-one-out)

OOB samples

Bootstrapping step (reminder)

- **Input**: a data set $\mathcal{D}_n$ with 8 observations

$\mathcal{D}_n$	obs 1	obs 2	obs 3	obs4	obs 5	obs 6	obs 7	obs 8

- **Do bootstrapping**: Draw observations randomly and with replacement into $\mathcal{D}_n$. Repeat independently this procedure B times.

$\mathcal{D}_n^1$	obs2	obs 6	obs 5	obs 5	obs 2	obs 4	obs 8	obs 7
$\mathcal{D}_n^2$	obs 6	obs 8	obs 2	obs 4	obs 2	obs 7	obs 8	obs 1
				$\vdots$				
$\mathcal{D}_n^B$	obs 7	obs 8	obs 3	obs 7	obs 2	obs 3	obs 3	obs 1

➜ **Output**: B bootstrap-resampled versions of the original $\mathcal{D}_n \Leftrightarrow$ B "new" data sets made up of observations of $\mathcal{D}_n$

$$\mathcal{D}_n^1, \ldots, \mathcal{D}_n^B.$$

OOB samples

Bootstrapping step

- **Input**: a data set $\mathcal{D}_n$ with 8 observations

$\mathcal{D}_n$	obs 1	obs 2	obs 3	obs4	obs 5	obs 6	obs 7	obs 8

- **Do bootstrapping**: obtain B bootstrap samples

$\mathcal{D}_n^1$	obs2	obs 6	obs 5	obs 5	obs 2	obs 4	obs 8	obs 7
$\mathcal{D}_n^2$	obs 6	obs 8	obs 2	obs 4	obs 2	obs 7	obs 8	obs 1
				$\vdots$				
$\mathcal{D}_n^B$	obs 7	obs 8	obs 3	obs 7	obs 2	obs 3	obs 3	obs 1

- B **OOB samples**: observations in $\mathcal{D}_n$ that are not in the B bootstrap samples.

OOB^1 :	obs 1	obs 3	obs 6	obs 6	obs 6	obs 6	obs 6	ind 6
OOB^2 :	obs 2	obs 5	obs 6					
				$\vdots$				
OOB^B :	obs 4	obs 5	obs 7					

➜ Use the OOB samples to estimate the error of a random forest.

OOB error

Notations:

- $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$,
- f_n^{RF}: the prediction function of a RF fitted on $\mathcal{D}_n$.

OOB error

For each couple $(\mathbf{X}_i, Y_i)$ in $\mathcal{D}_n$, compute its RF predictor by averaging only the trees associated to boostrap samples that do not included $(\mathbf{X}_i, Y_i)$:

$$f_n^{RF}(\mathbf{X}_i) = \frac{1}{\sharp \mathcal{I}_i^B} \sum_{b \in \mathcal{I}_i^B} f_n^b(\mathbf{X}_i),$$

where $\mathcal{I}_i^B$ denotes the set of indices for boostrap samples for which $(\mathbf{X}_i, Y_i)$ is out-of-bag.

In regression: $f_n^{RF}(\mathbf{X}_i) = \frac{1}{\sharp \mathcal{I}_i^B} \sum_{b \in \mathcal{I}_i^B} (Y_i - f_n^{RF}(\mathbf{X}_i))^2$.

In classification: $f_n^{RF}(\mathbf{X}_i) = \frac{1}{\sharp \mathcal{I}_i^B} \sum_{b \in \mathcal{I}_i^B} \mathbb{1}_{Y_i \neq f_n^{RF}(\mathbf{X}_i)}$.

Content

Introduction: model averaging

Bagging

Random forests

- Algorithm
- Out-Of-Bag sample
- **Variable importance**

Model interpretation

One major asset of the single trees: models highly interpretable.

A random forest = aggregation of several single trees.
➜ Not directly interpretable.

Alternative with RF: introduction of an importance score that measures the prediction strength of each feature.
➜ Importance score used OOB samples.
➜ Importance score can be used to rank features and so to perform features selection.

Variable importance

1) Compute the OOB error of each tree.

- OOB_b = the OOB sample of the b-th tree T^b
- E_{OOB_b} = the OOB error of T^b on OOB_b.
  ➜ In regression: $E_{OOB_b} = \frac{1}{\sharp OOB_b} \sum_{i \in OOB_b} (Y_i - f_n^b(\mathbf{X}_i))^2$ (quadratic loss),

  ➜ In classification: $E_{OOB_b} = \frac{1}{\sharp OOB_b} \sum_{i \in OOB_b} \mathbf{1}_{f_n^b(\mathbf{X}_i) \neq Y_i}$ (misclassification error).

Variable importance

2) Permute randomly values of the jth feature and compute the OOB error on the permuted sample.

- OOB_b^j = the bth OOB sample with the jth feature permuted.
- $\mathbf{X}_i^j$ = the ith observation in OOB_b^j with permuted value for input j.
- $E_{OOB_b^j}$ = the error of T^b on OOB_b^j.

  ➜ In regression: $E_{OOB_b^j} = \frac{1}{\sharp OOB_b^j} \sum_{i \in OOB_b^j} (Y_i - f_n^b(\mathbf{X}_i^j))^2$ (quadratic loss),

  ➜ In classification: $E_{OOB_b^j} = \frac{1}{\sharp OOB_b^j} \sum_{i \in OOB_b^j} \mathbf{1}_{f_n^b(\mathbf{X}_i^j) \neq Y_i}$ (misclassification error).

Variable importance

3). Compute the importance score for the *j*th feature

$$VI(\mathbf{X}^j) = \frac{1}{B}\sum_{b=1}^{B}(E_{OOB_b^j} - E_{OOB_b})$$

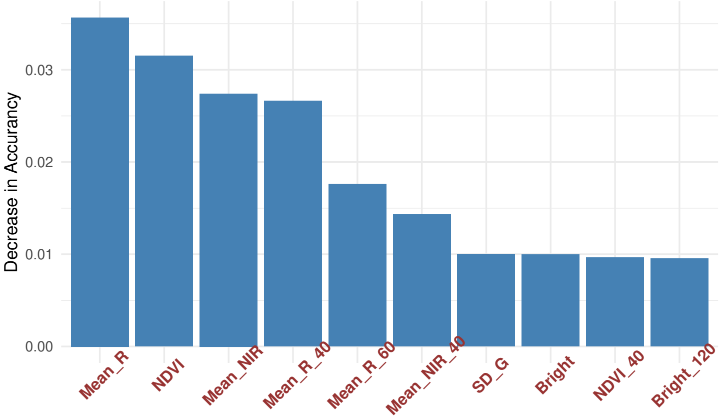
- **Meaning**: the importance score of the *j*th feature is defined as the average over the B trees of the difference between the OOB error and the permuted.

- **Explanation**: Permutations are used to mimics independence (no link) between $\mathbf{X}^j$ and Y.
  ➜ The feature $\mathbf{X}^j$ is considered as important if by breaking the link between $\mathbf{X}^j$ and Y, the error on the OOB samples increase.

- **Interpretation**: Large difference between the OOB error and the permuted OOB error ⇒ large $VI(\mathbf{X}^j)$ ⇒ $\mathbf{X}^j$ is an important feature to predict Y.

Variable importance: example on the land cover data set

The 10 variables with the highest importance score (see TP2):



Conclusion

- Random Forest = an improvement over bagged CART trees.
- Algorithm with few paramters to tune.
- No overfitting problem compared to a single trees.
- Higher performances than a single CART tree or bagged CART trees.
- No need of test set: out-of-bag samples.
- Interpretation tool: variable importance score.
- Large applicability.

References

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

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

Freund, Y., Schapire, R. E., et al. (1996).
Experiments with a new boosting algorithm.
In *icml*, volume 96, pages 148–156. Citeseer.