A. Giovanidis 2019

10. Tree-based methods

Data Analysis for Networks - DataNets'19
Anastasios Giovanidis

Sorbonne-LIP6

October 16, 2019

Bibliography

B.1 Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. "An introduction to statistical learning: with applications in R". Springer Texts in Statistics.

☞ Chapter 7

ISBN 978-1-4614-7137-0 (DOI 10.1007/978-1-4614-7138-7)

B.2 Andy Liaw and Matthew Wiener. "Classification and Regression by randomForest". R News, Vol. 2/3, December 2002.

Trees

A. Giovanidis 2019

Tree-based methods: The training set is split into a number of simple regions.

- The splits are done based on some splitting rules over the features.
- They are used both for regression and classification.
- Trees are **simple and useful for interpretation**.
- Worse in performance compared with other supervised learning approaches.

☞ Improved tree-based methods: bagging, random forests, boosting.

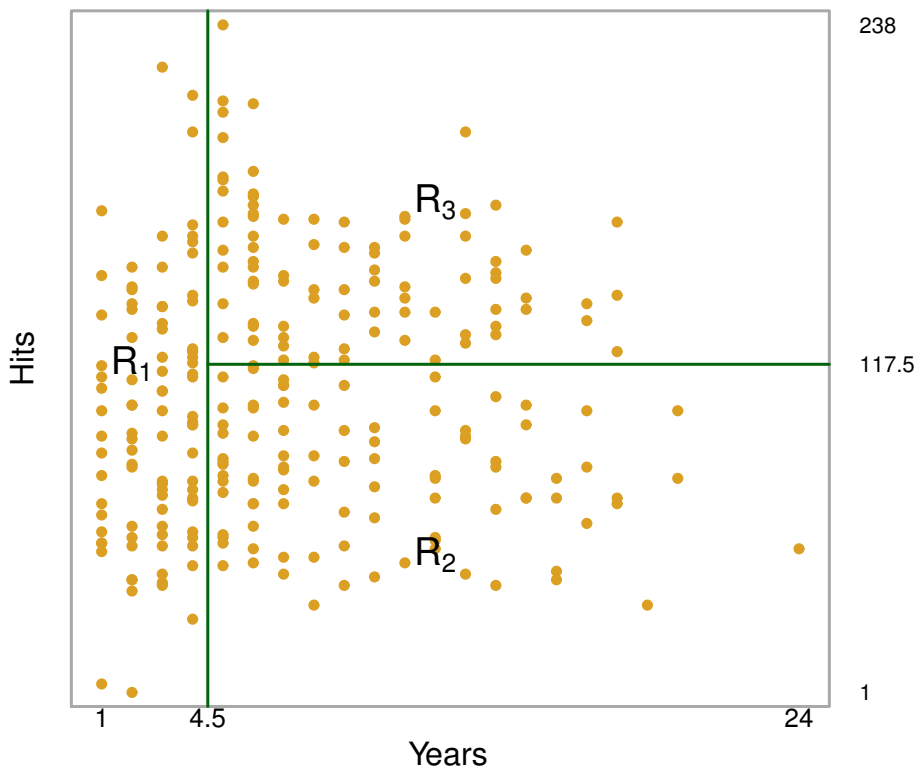
Regression Trees

Tree Regression: An example (1)

A. Giovanidis 2019

Tree Regression: An example (2)

A. Giovanidis 2019



A. Giovanidis 2019

Notations

- The regions R_1, R_2 and R_3 are known as terminal nodes or leaves of the tree.
- The points of split are called internal nodes.
- The segments of the tree "departing" from the nodes are called branches.
- At a terminal node, all data entries who satisfy the relevant rules are **averaged**.

☞ In the example above, the most important feature is shown at the first node (Years) and then it follows (Hits) as secondary criterion to differentiate between all players with experience more than 4.5 years.

A. Giovanidis 2019

Building a regression tree

Each sample (X_i, y_i) has $p \geq 1$ predictors and 1 outcome.
We will call the space of X the predictor space and is p-dimensional.

Step.1 Divide the predictor space into $M > 1$ **distinct** and **non-overlapping** regions $R_1,\ R_2, ...,\ R_M$.

Step.2 For every observation that falls into the region R_m we make the same prediction equal to:

the mean of all the response training values inside R_m.

e.g. If $M = 2$ and for $R_1 : \hat{y}_1 = 10$, for $R_2 : \hat{y}_2 = 20$ then every X that falls on R_1 will get the prediction 10.

How to construct the regions?

A. Giovanidis 2019

Each Region corresponds to a high dimensional rectangle (**box**).

Find boxes $R_1, \ldots, R_M$ that minimize the RSS

$$\sum_{j=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2.$$

- $\hat{y}_{R_m}$ is the mean response for the training observations in the m-th box.

☞ Very costly to consider every possible partition into M boxes!

Top-down Greedy

Take a top-down approach : recursive binary splitting.

- Begin at the top of the tree.
- Successively split the predictor space.
- Each split is indicated via two new branches further down the tree.
- The best split is made at the particular step (**Greedy**).

A. Giovanidis 2019

Recursive binary splitting

Select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions

$$R_1(j,s) = \{X | X_j < s\} \quad \& \quad R_2(j,s) = \{X | X_j \geq s\},$$

leads to the greatest possible reduction in RSS.

$$\min_{(j,s)} \quad \sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Next we split the process further for the remaining points in each region.

Stopping criterion

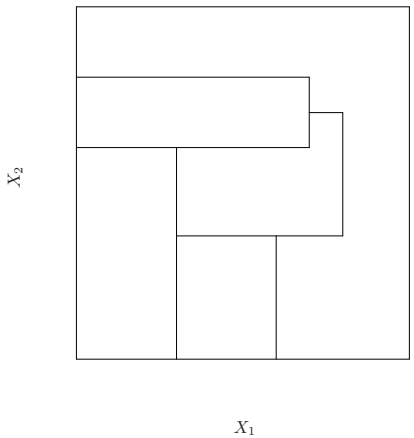
When do we stop? How many regions do we need?

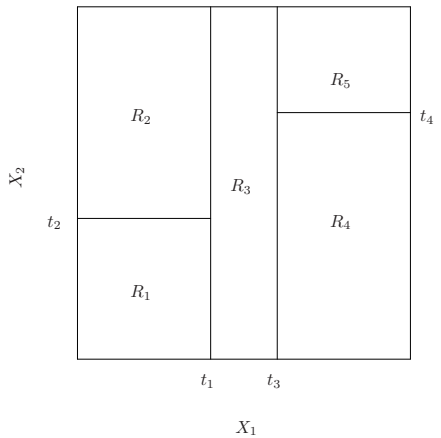
Stopping criteria:

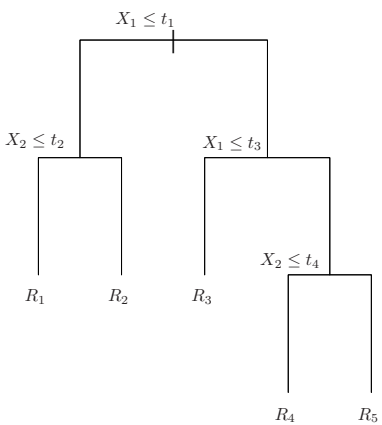
- Stop when no region contains more than $\alpha = 5$ observations.
- (or) Stop when the depth of the tree is no more than $d = 3$ levels.

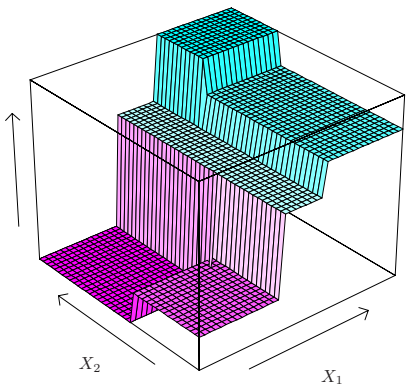
Another example with 5 regions

A. Giovanidis 2019









Tree pruning (aka Regularisation)

A. Giovanidis 2019

When the tree is very large with lots of leaves, it describes exactly the training data, but it can lead to overfitting for the test data.

A solution is to grow the large tree T_o first and then prune it!

$$\min_{|T|} \quad \sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

given $\alpha \geq 0$.

For $\alpha = 0$ then the optimal is T_o.
For $\alpha >> 1$ then the optimal tree has depth 0 (no split).
In-between values of α correspond to pruned trees T.
Pick the one for which the test error is minimised by cross-validation.

General Method

A. Giovanidis 2019

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

A. Giovanidis 2019

Classification Trees

Classification

A. Giovanidis 2019

A classification tree is used to predict a qualitative response (rather than a quantitative one).

☞ In such trees, each response belongs to the most commonly occurring class of training observations in the region to which it belongs.

Not only the prediction of a class is important, but also the class proportions among training observations that fall in this region.

We use recursive binary split, again.

A. Giovanidis 2019

Mis-classification Cost

- Classification error rate: the fraction of training observations in that region m that do not belong to the most common class,

$$E_m = 1 - \max_k(\hat{p}_{m,k}).$$

- Gini Index: a measure of total variance across the K classes,

$$G_m = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}).$$

  It takes a small value if all the $\hat{p}_{m,k}$ are close to zero or one.

- Entropy: similar in behavior to Gini index

$$D_m = -\sum_{k=1}^{K} \hat{p}_{m,k} \log \hat{p}_{m,k}.$$

Use of measures

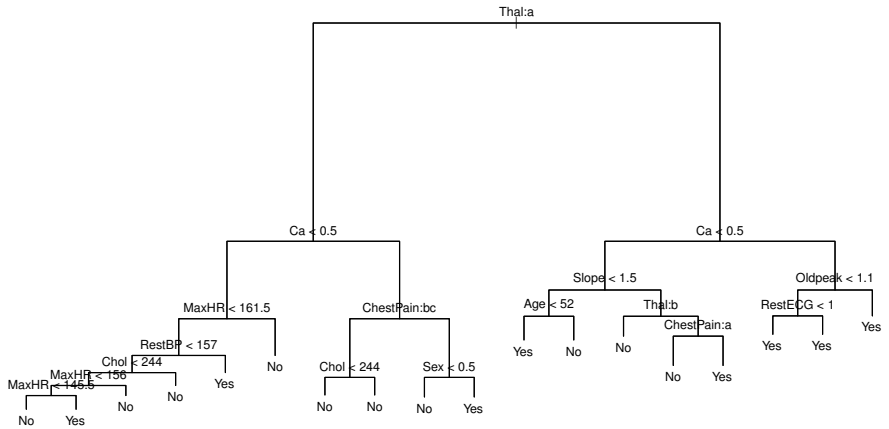
A. Giovanidis 2019

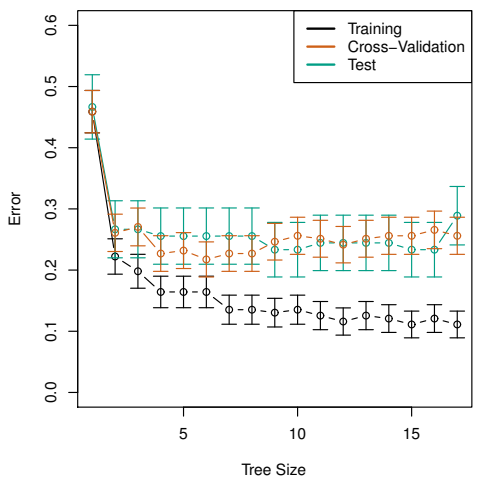
Gini, Entropy: sensitive to **node purity in a region**.

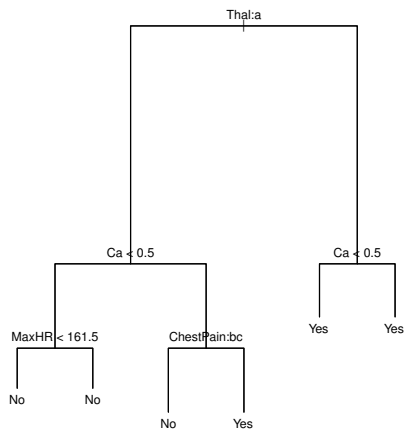
- Use either Gini index or Entropy to evaluate **quality of a split**.
- If **prediction accuracy** of the total tree is the goal, use classification error rate.
- For **pruning**, any of these measures can do.

Classification example

A. Giovanidis 2019







A. Giovanidis 2019

Exercise

From data in the Table, construct a decision tree as follows:

i. at each step compute the Gini index for every possible split considering all attributes and select the split with the best Gini value;

ii. Stopping rule: when the Gini value of a node is zero or no further split is possible; and

iii. The class of a leaf node is determined by a majority rule.

IoT device	Delay-tolerant (d)	Throughput (t)	**Class**
α_1	Y	H	A
α_2	N	H	A
α_3	N	H	B
α_4	Y	L	A
α_5	N	L	B

A. Giovanidis 2019

Solution

$\#Features = 2$: $\{d,\ t\}$
$\#Classes = 2$: $\{A,\ B\}$

- First check feature d.

YES: $\{\alpha_1(A),\ \alpha_4(A)\}$, majority class: A.
NO: $\{\alpha_2(A),\ \alpha_3(B),\ \alpha_5(B)\}$, majority class: B.
Gini(d)
$= [2/2 \cdot (1 - 2/2) + 0 \cdot (1 - 0)] + [2/3 \cdot (1 - 2/3) + 1/3 \cdot (1 - 1/3)] =$
$4/9$.
Error(d) $= [1 - 1] + [1 - 2/3] = 1/3$.

- Then check feature t.

HIGH: $\{\alpha_1(A),\ \alpha_2(A),\ \alpha_3(B)\}$, majority class: A.
LOW: $\{\alpha_4(A),\ \alpha_5(B)\}$, majority class: A or B.
Gini(t) $= 2 \cdot 2/3 \cdot 1/3 + 2 \cdot 1/2 \cdot 1/2 = 17/18$.
Error(t) $= 1/3 + 1/2 = 5/6$.
Hence the split by feature d has a smaller Gini index and Error.

Solution cont'd

A. Giovanidis 2019

For $\{d : YES\}$ the majority class is A with node purity, thus no further split is necessary.

For $\{d : NO\}$ the training data $\{\alpha_2(A),\ \alpha_3(B),\ \alpha_5(B)\}$ will be either split or not.

- If they get split by feature t:
  HIGH: $\{\alpha_2(A),\ \alpha_3(B)\}$, majority class: A or B.
  LOW: $\{\alpha_5(A)\}$, majority class: B.
  Gini(d:YES, t) $= 2 \cdot 1/2 \cdot 1/2 + 0 = 1/2$.
  Error(d:YES, t) $= 1/2$.

☞ But $1/2 > 4/9$ so no further split for the tree based on feature (t)!

Trees VS Linear Models

A. Giovanidis 2019

For linear regression the model assumed is of the form

$$f(X) \quad = \quad \beta_0 + \sum_{j=1}^{p} X_j \beta_j,$$

whereas regression trees assume a model of the form

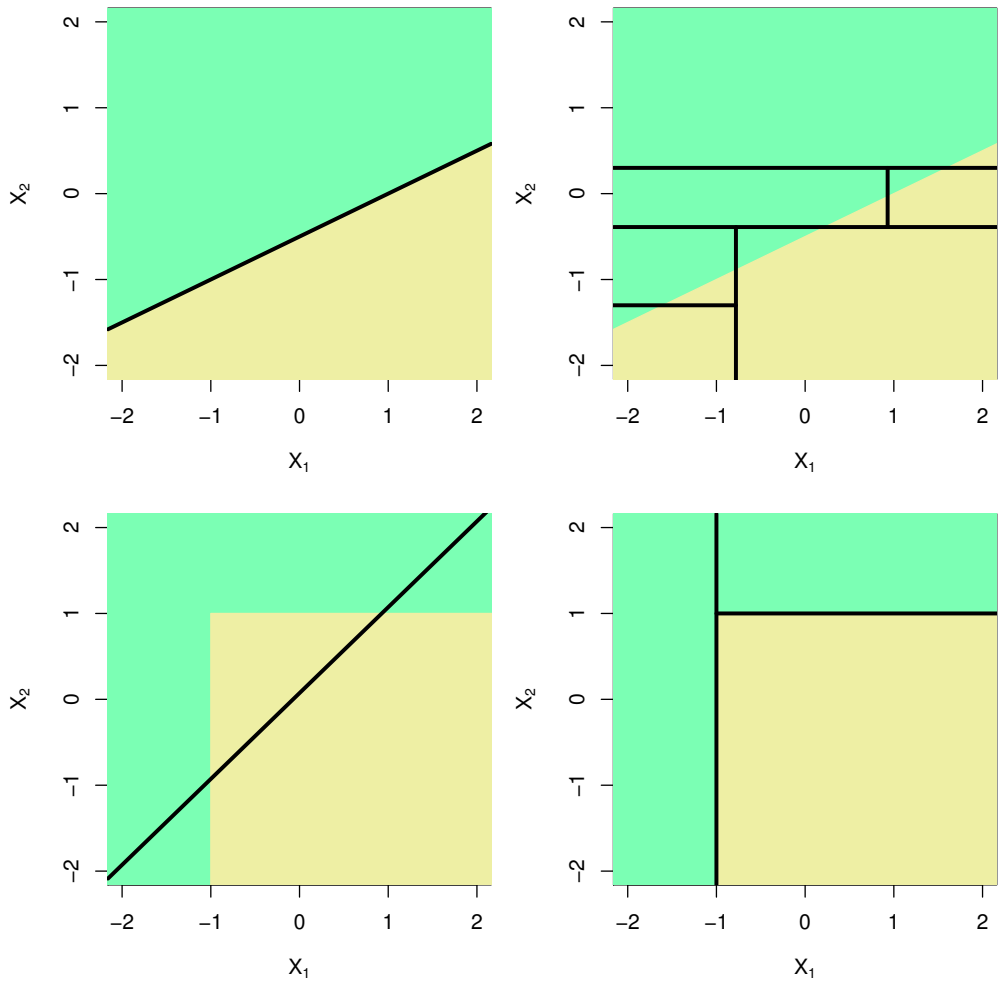
$$f(X) \quad = \quad \sum_{m=1}^{M} c_m \cdot \mathbf{1}_{(X \in R_m)},$$

where $R_1, \ldots, R_M$ represent a partition of the feature space.

Which one is better?

Trees VS Linear Models cont'd

A. Giovanidis 2019



Decision Trees: Pros and Cons

A. Giovanidis 2019

Pros:

- Easy to explain to others.
- Mirror human decision-making.
- Graphical Display.

Cons:

- Do not have a high level of accuracy.
- Can be very non-robust, i.e. a small change in the data can create a large change in the resulting tree.

A. Giovanidis 2019

Bagging, Random Forests, Boosting

Variance reduction techniques

A. Giovanidis 2019

Decision trees have high-variance: two different training sets from the same sample can give two completely different trees.

☞ Bootstrap aggregation or bagging reduces the variance of the method.

Idea: Use B separate prediction models, each using a different training set, and **average** the resulting predictions (*for regression*)

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x).$$

Problem: Where do we find multiple training sets?

Bagging

A. Giovanidis 2019

Bagging is based on an old idea: bootstrap the single training data-set!

- Generate B different bootstrapped training data sets.
- Train the method on each bootstrapped training set to get $\hat{f}^{*b}(x)$,
- Finally, average over all the predictions (*for regression*).

Bagging does not include pruning: individual trees grow deep with high variance but low bias. Averaging these B trees reduces the variance.

Bagging for Classification

A. Giovanidis 2019

In classification we cannot average because we have qualitative response.

- Generate B different bootstrapped training data sets.
- Train the method on each bootstrapped training set to get $\hat{f}^{*b}(x)$,
- Here, $\hat{f}^{*b}(x)$ is the class predicted by tree b.
- Take a majority vote: the most occurring class among the B predictions.

Out-Of-Bag Error Estimation

A. Giovanidis 2019

☞ No need to perform cross-validation in bagged models.

- In bagging, trees are repeatedly fit to bootstrapped subsets of the observations.
- On average, each bagged tree makes use of 2/3's of the observations.
- We can predict the response for the i-th observation using each of the trees in which the observation was Out-Of-Bag (OOB).
- The process will yield about $B/3$ predictions, and we can average (*regression*) or take a majority vote (*classification*).

The overall OOB MSE or classification error can be computed.

☞ With B sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error.

Interpretability

A. Giovanidis 2019

☞ Bagging improves prediction accuracy at the cost of interpretability.

✏ We can obtain a summary of importance of each predictor using the RSS (*regression*) or the Gini index (*classification*).

✏ We average over all B trees the amount that the RSS or Gini decreases by splits over a given predictor.

Random Forests

A. Giovanidis 2019

☞ Problem: Predictions from bagged trees can be highly correlated!
e.g. a strong predictor in the data will always be chosen for the top split.

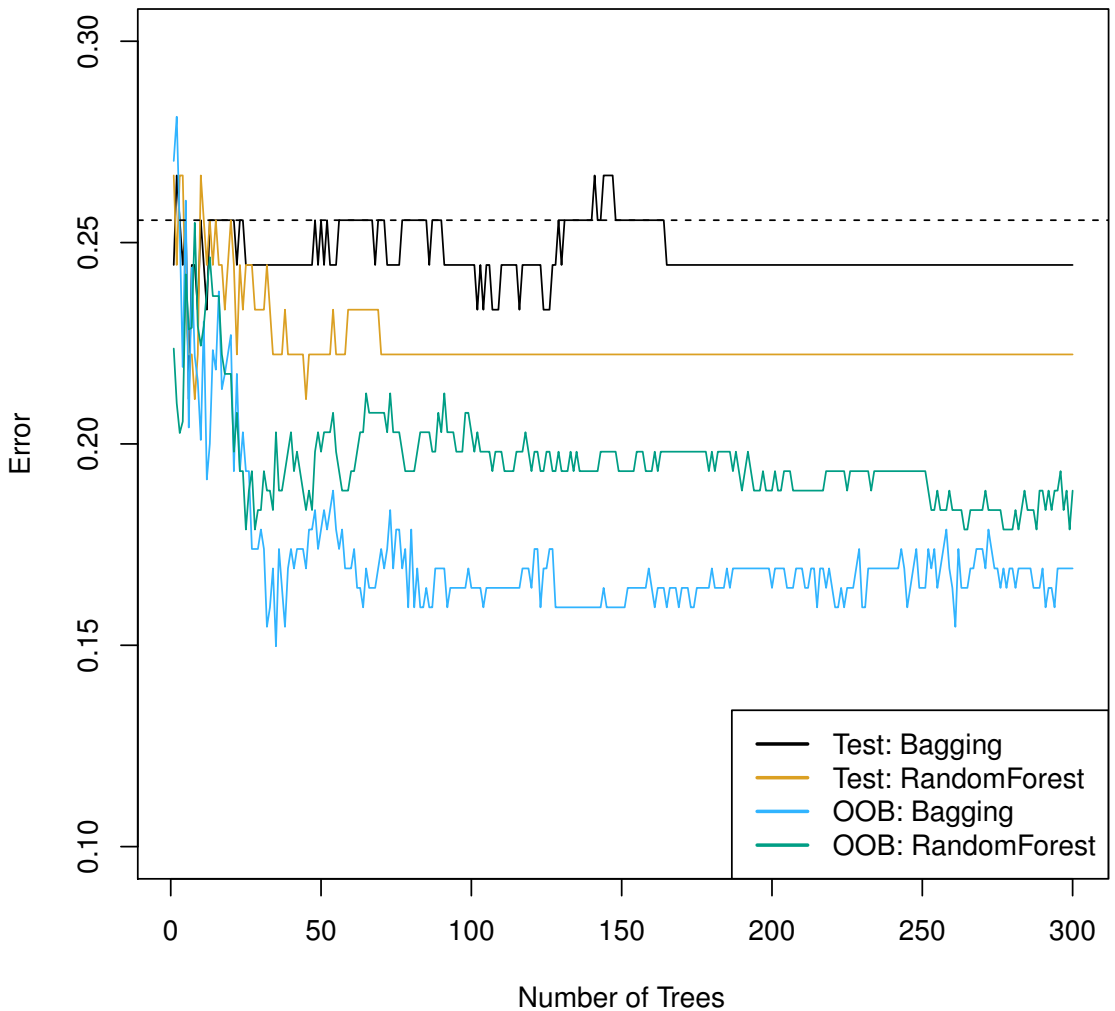
How to decorrelate?

Random forest: For each bootstrapped set, at each split in the tree, **the algorithm considers a random subset of the p totally available features / predictors!**

Design Hint: At each split, we choose $m \approx \sqrt{p}$ predictors randomly. So if $p = 9$ features, we decide among $m = 3 = \sqrt{9}$ features per split.

Random Forests VS Bagging I

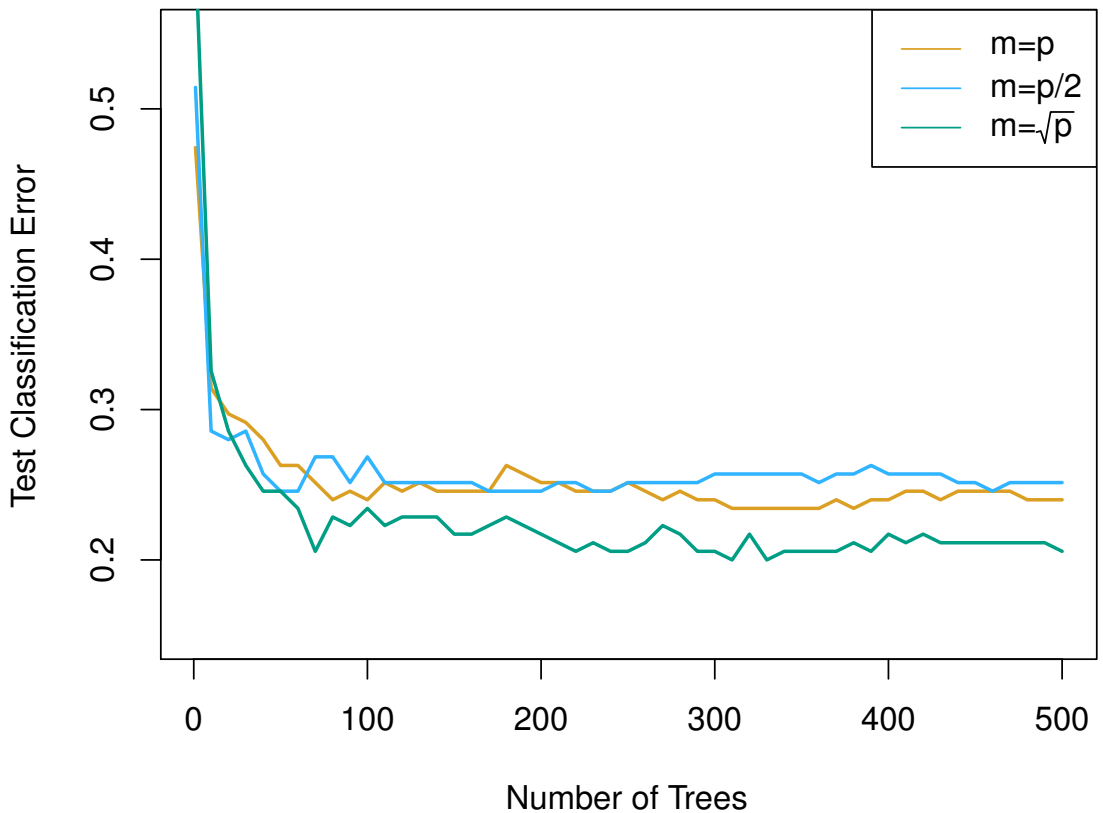
A. Giovanidis 2019



A. Giovanidis 2019

Random Forests VS Bagging II

If $m = p$ per split, then the Random Forest corresponds just to Bagging, otherwise, the improvement is visible.



Boosting

A. Giovanidis 2019

As in Bagging, Boosting also involves growing several decision trees.

> But, contrary to Bagging where trees are independent,
> Boosting grows trees sequentially.
> ☞ Each tree is grown using information from previous trees.

Each tree is fit on a modified version of the original data set.

In Boosting, the model *learns slowly*.

A. Giovanidis 2019

Boosting parameters

1. The number of trees B.
2. The shrinkage parameter $\lambda > 0$.
3. The number $d \geq 1$ of splits in the tree.

A. Giovanidis 2019

Boosting algorithm

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

A. Giovanidis 2019

END