Random Forest for Classification Problems

Raphael, Arkadiusz and Burak

Uni Bonn

January 15, 2020

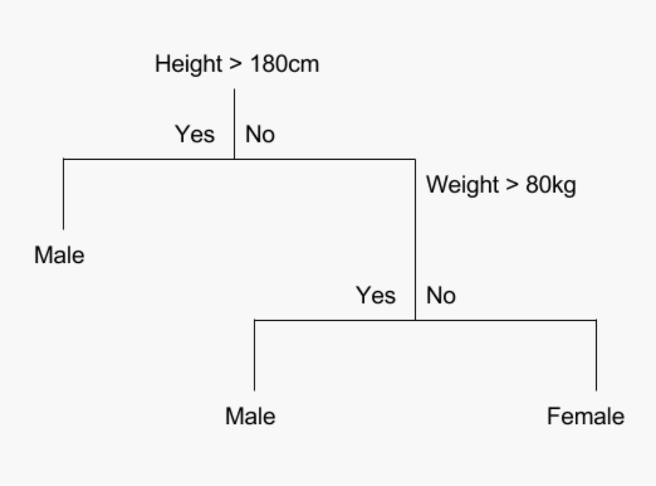
Overview

Intro

Sed iaculis dapibus gravida. Morbi sed tortor erat, nec interdum arcu. Sed id lorem lectus. Quisque viverra augue id sem ornare non aliquam nibh tristique. Aenean in ligula nisl. Nulla sed tellus ipsum. Donec vestibulum ligula non lorem vulputate fermentum accumsan neque mollis.

Sed diam enim, sagittis nec condimentum sit amet, ullamcorper sit amet libero. Aliquam vel dui orci, a porta odio. Nullam id suscipit ipsum. Aenean lobortis commodo sem, ut commodo leo gravida vitae. Pellentesque vehicula ante iaculis arcu pretium rutrum eget sit amet purus. Integer ornare nulla quis neque ultrices lobortis. Vestibulum ultrices tincidunt libero, quis commodo erat ullamcorper id.

Decision Tree: Example



Decision Tree: Tree Building Process

A tree is grown starting from the root node by repeatedly using the following steps on each node (also called binary splitting):

(i) **Find best split s for each feature X_m:** For each feature X_m, there exist $K - 1$-many potiential splits whereas K is the number of different values for the respective feature. Evaluate each value $X_{m,i}$ at the current node t as a candidate split point (for $x \in X_m$, if $x \leq X_{m,i} = s$, then x goes to left child node t_L else to right child node t_R). The best split point is the one that maximize the splitting criterion $\Delta i(s,t)$ the most when the node is split according to it. The different splitting criteria will be covered in the next chapter.

(ii) **Find the node's best split:** Among the best splits for each feature from Step (i) find the one s^*, which maximizes the splitting criterion $\Delta i(s,t)$.

(iii) **Satisfy stopping criterion:** Split the node t using best node split s^* from Step (ii) and repeat from Step (i) until stopping criterion is satisfied.

Decision Tree: Purity Measures

Gini Measure

$$i(t) = \sum_{c \in C} p(c|t)(1 - p(c|t)) = 1 - \sum_{c \in C} p_c^2 \tag{1}$$

Information Entropy

$$i(t) = \sum_{c \in C} p(c|t) log(p(c|t)) \tag{2}$$

Expected Generalization Error

Let $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ and $y = f(x) + \epsilon$.
The decomposition of a model's expected generalization error is

$$\boldsymbol{Err}(f(x)) = Noise(x) + [Bias(\hat{f}(x))]^2 + Var(\hat{f}(x)) \tag{3}$$

Noise is irreducible and independent of the model.
There is a trade-off between $bias^2$ and variance.
Adjusting parameters to decrease variance increases $bias^2$ and vice versa.

Bias-Variance Trade-off

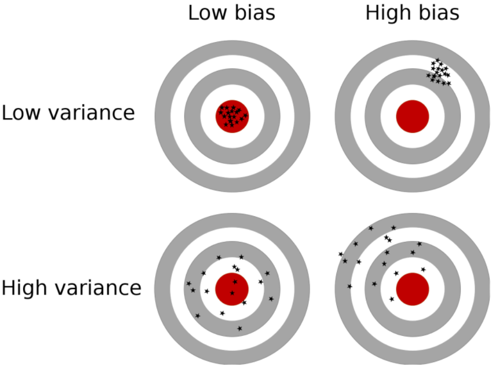
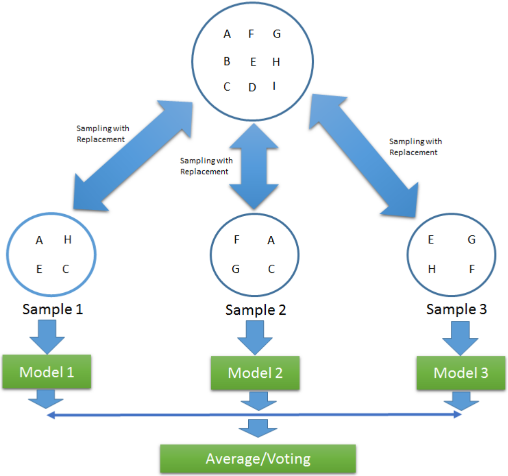


Figure: Illustration of bias-variance trade-off [2]

Decision trees generally have low bias and high variance [1].

Bagging

1. created for methods with high variance
2. reduces variance and gives better predictions
3. improvement of bagging: Random Forest



Random Forest

An ensemble of randomly trained decision trees, so in other words random forest was defined by L. Breiman:

Theorem

A random forest is a classifier consisting of a collection of tree-structured classifiers $\hat{T}_{\theta_b}(\mathbf{x}), b = 1, ..., B$ where the θ_b are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$.

Random Forest is an extension and improvement over bagging:

1. Like in bagging, multiple decision trees are built
2. Improvement: an injection of randomness is made

Random Forest: randomness in the model

Two key concepts that makes decision forest "random" are:

1. Random sampling of training data points when building trees
2. Random subsets of features considered when splitting nodes. Recommended number of variables:
   a. For classification: $\lfloor \sqrt{n} \rfloor$
   b. For regression: $\lfloor \frac{n}{3} \rfloor$

Random Forest: algorithm

Algorithm 1: Random Forest for Regression or Classification

1. For $b = 1$ to B:
   a. Draw a bootstrap sample θ_b of size N from the training data.
   b. Grow the Random Forest tree T_{θ_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 n 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 $\{T_{\theta_b}\}_1^B$

Mathematical Concept

Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

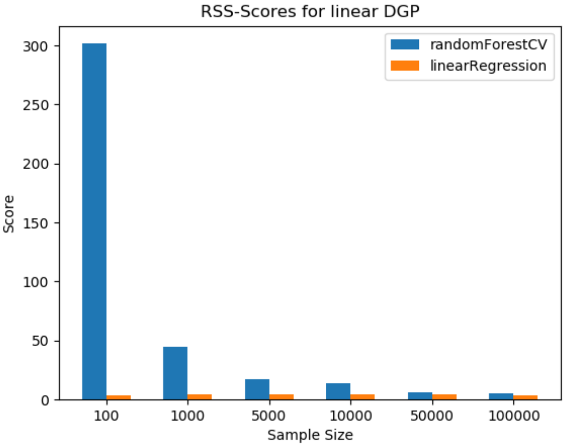
Simulation Study: Linear DGP

The linear DGP generates the data tuples (y, x_1, x_2, x_3) as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon, \tag{4}$$

whereas $(\beta_0, \beta_1, \beta_2, \beta_3) = (0.3, 5, 10, 15)$, $x_1, x_2, x_3 \sim \mathcal{N}(0,\ 3)$, and $\epsilon \sim \mathcal{N}(0,\ 1)$.

Decision Tree: Linear DGP Results



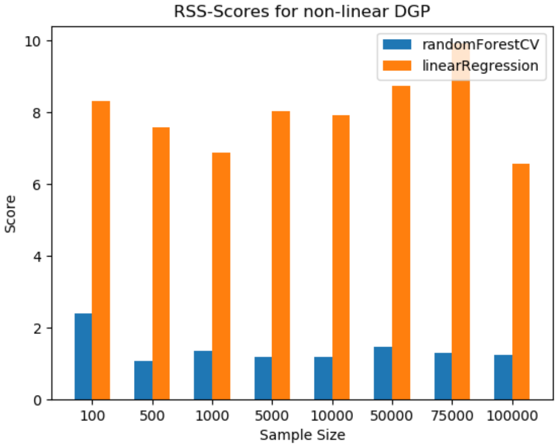
Simulation Study: Non-Linear DGP

The non-linear DGP generates the data tuples (y, x_1, x_2) as follows:

$$y = \beta_0 + \beta_1 \mathbb{1}(x_1 \geq 0, x_2 \geq 0) + \beta_2 \mathbb{1}(x_1 \geq 0, x_2 < 0) + \beta_3 \mathbb{1}(x_1 < 0) + \epsilon, \quad (5)$$

whereas $(\beta_0, \beta_1, \beta_2, \beta_3)$, x_1, x_2 and ϵ are the same as in the previous DGP.

Simulation Study: Non-Linear DGP Results



Real Data

Data: Titanic data

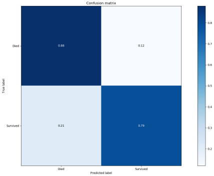
Method used:

1. Random Forest
2. AdaBoost
3. Gradient Boosting Classifier

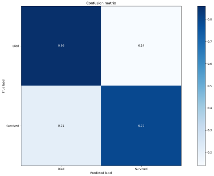
Goal: Given features of passengers predict which passengers survived the Titanic shipwreck

Real Data: results

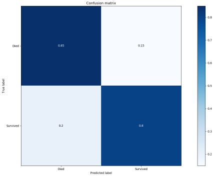
Random Forest
Accuracy: 84,32%



AdaBoost
Accuracy: 82.8%



Gradient Boosting
Accuracy: 82,8%



The End

References

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

Daan van der Valk and Stjepan Picek. *Bias-variance decomposition in machine learning-based side-channel analysis*. Tech. rep. Cryptology ePrint Archive, Report 2019/570, 2019.