Understanding Random Forests

Mohamed Ndaoud

Performance of a binary classifier

Confusion Matrix

		Predictive condition	
		Positive	Negative
True condition	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

Confusion Matrix

- The sensitivity, or true positive rate (TPR), is:

$$TPR = \frac{TP}{TP + FN}.$$

- The specificity, or false positive rate (FPR), is:

$$FPR = \frac{FP}{FP + TN}.$$

- The precision, or the positive predictive value (PPV), is:

$$PPV = \frac{TP}{TP + FP}.$$

- The accuracy (ACC) is:

$$ACC = \frac{TP + TN}{TP + FN + FP + TN}.$$

Receiver Operating Characteristic

- The ROC (Receiver Operating Characteristic) curve plots the sensitivity (TPR) in function of the specificity (FPR) for different decision thresholds.

- The AUC (Area Under the ROC) measures how well a decision rule can classify:
  $AUC = 1/2$: worse case,
  $AUC = 1$: best case.

Decision Trees

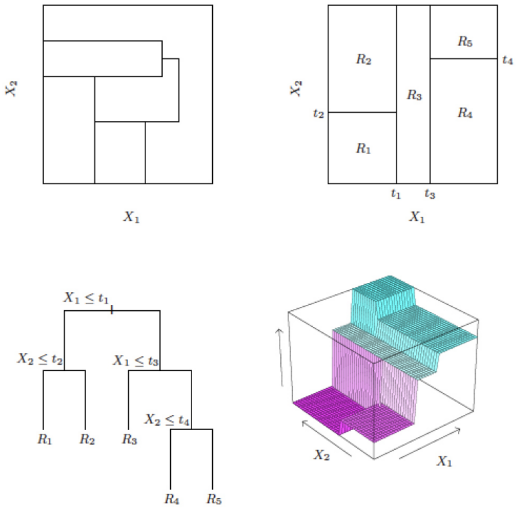
Principles

A Classification And Regression Tree (CART) is a recursive method:

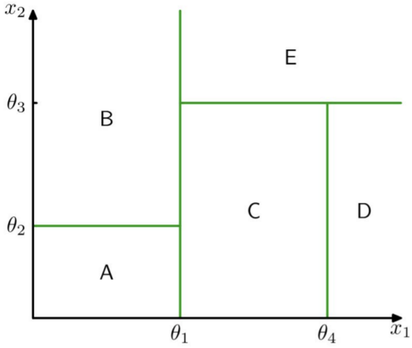
- At the root is the whole sample.
- Each node of the tree separates the sample into 2 branches, according to a discrete, continuous or ordinal variable (threshold) or a nominal variable (set of categories).
- A terminal node is called leaf.

We obtain a partition of the feature space into rectangles (recursive binary partitions), and then fit a simple model (average, majority) in each rectangle.

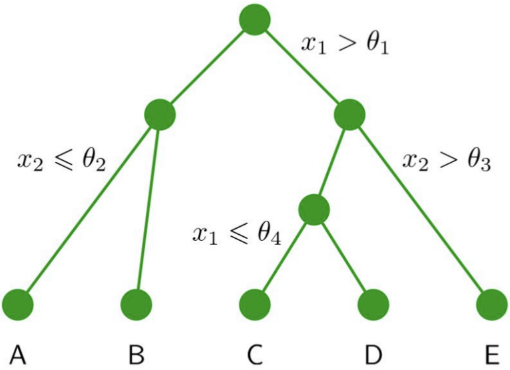
Example (Hastie et al., 2009)



Example 2

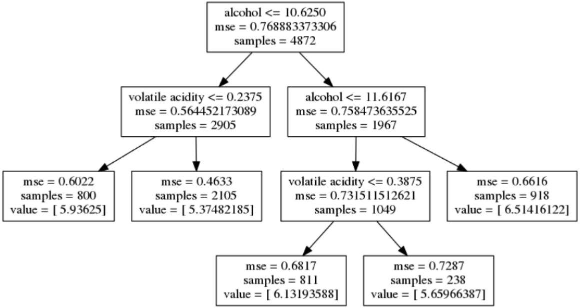


Example 2 - Continued



Tree visualization

```
# Display tree
from sklearn.tree import export_graphviz
export_graphviz(estimator, out_file="tree.dot",
                feature_names=feature_names)
```



Warning for categorical variables

- The algorithm tends to favor variables with many categories.
- It is recommended to reduce this number by merging certain categories.

Technical issues

- Choose the best split for each variable.
- Choose the best variable to split.
- Decide that a node is a leaf.
- Fit a simple model in each rectangle.

Overfitting

- A very large tree might overfit the data.
- A too small tree might not capture the underlying structure.

Classification trees

Goal

- Classify a categorical variable Y with K classes:

$$\{1, \dots, K\}.$$

- Based on p predictors:

$$(X_1, \dots, X_p).$$

Region forecast

Let consider a partition into M regions $\{R_1, \dots, R_M\}$.
Let C_m be the class of the $m-$th region.
For $k \in \{1, \dots, K\}$, we estimate $P(C_m = k)$ by :

$$\hat{p}_k^m = \frac{1}{Card(x_i \in R_m)} \sum_{x_i \in R_m} \mathbf{1}(y_i = k).$$

The predicted class for the $m-$th region is the most important class among points in the region:

$$\hat{c}_m = \arg \max_{k \in \{1,\dots,K\}} \hat{p}_k^m.$$

Splitting criterion

To find the best binary partition with a sum of squares criterion, we use a "greedy" algorithm.
Let consider a binary partition for the $j-$th predictor and a split point s:

$$R_1(j,s) = \{X/X_j \leq s\},$$
$$R_2(j,s) = \{X/X_j > s\}.$$

We choose the splitting variable j and the split point s that solve (minimization of the missclassification errors):

$$\min_{j\in\{1,\ldots,p\}} \min_s (1-\hat{p}^m_{\hat{c}_1}) + (1-\hat{p}^m_{\hat{c}_2}).$$

Once we have found this optimal split, we repeat the splitting step on the two regions obtained, and so on ...

Impurity function

We have used an impurity function i: the missclassification error. More generally, we consider functions such that:

- i is minimal, and equal to 0, for configurations with only one class:

$$(1, 0, 0, \ldots, 0)$$

$$(0, 1, 0, \ldots, 0)$$

$$\vdots$$

$$(1, 0, 0, \ldots, 0).$$

- i is maximal for the configuration:

$$\forall i \in \{1, \ldots, K\} : p_i = \frac{1}{K}.$$

Examples of impurity functions

For a region R_m:

- Misclassification error:

$$i(R_m) = (1 - \hat{p}^m_{\hat{c}_1}).$$

- Gini index:

$$i(R_m) = \sum_{k=1}^{K} \hat{p}^m_k (1 - \hat{p}^m_k).$$

- Cross-entropy:

$$i(R_m) = -\sum_{k=1}^{K} \hat{p}^m_k \log(\hat{p}^m_k).$$

Advantages and disadvantages of trees

Advantages

- No distribution assumption.
- Easy to implement.
- Nice graphical representation of a set of rules.
- Automatic variable selection.

Disadvantages

- Need large data sets.
- Only horizontal or vertical splits.
- No interactions between variables.
- Instability (a small change in the data set can provide a different tree).

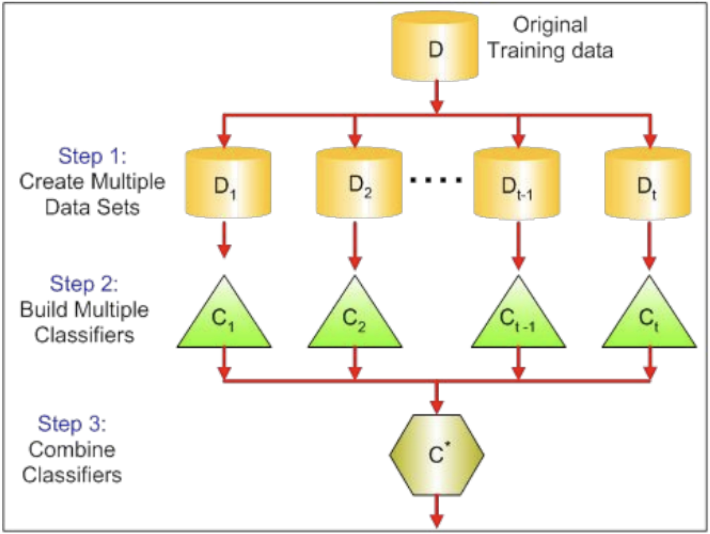
Model averaging

- Bagging: (Breiman, 1996)
  Fit many trees to bootstrap-resampled versions of the training data set, and "combine" them (average or majority vote).

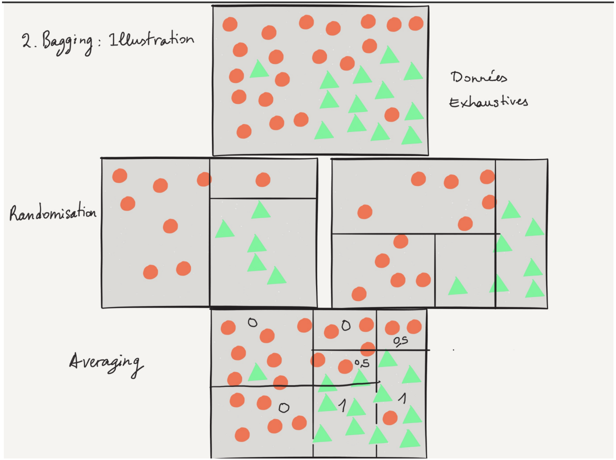
- Boosting: (Freund & Shapire, 1996)
  Fit many large or small trees to reweighted versions of the training data set, and "combine" them (with weights).

- Random Forests: (Breiman, 2001)
  Fit many de-correlated trees, and "combine" them.

Bagging example



Bagging illustration



Boosting illustration