Extra Trees vs Random Forest

Extra Trees and **Random Forest** are two very similar ensemble methods and often a doubt arises as to whether to use one or the other. What is really the difference between them?

In previous posts, , we have seen how to create a Random Forest from decision trees and how they improve their performance. Furthermore, if you want more information about simple decision trees, you can find read "".

Therefore, **trees ensemble methods are better than simple decision trees**, but is an ensemble better than the other? Which one should I use? This post is going to try to answer these questions, studying the differences between the Extra Trees and Random Forest and comparing both of them in terms of results.

The two ensembles have a lot in common. Both of them are **composed of a large number of decision trees**, where the final decision is obtained taking into account the prediction of every tree. Specifically, by majority vote in classification problems, and by the arithmetic mean in regression problems. Furthermore, both algorithms have the same growing tree procedure (with one exception explained below). Moreover, when selecting the partition of each node, both of them randomly choose a subset of features.

So, the main two differences are the following:

- **Random forest uses bootstrap replicas**, that is to say, it subsamples the input data with replacement, whereas **Extra Trees use the whole original sample**. In the Extra Trees *sklearn* implementation there is an optional parameter that allows users to bootstrap replicas, but by default, it uses the entire input sample. This may increase variance

because bootstrapping makes it more diversified.

- Another difference is **the selection of cut points** in order to split nodes. **Random Forest chooses the optimum split while Extra Trees chooses it randomly**. However, once the split points are selected, the two algorithms choose the best one between all the subset of features. Therefore, Extra Trees adds randomization but still has optimization.

These differences motivate the reduction of both bias and variance. On one hand, using the whole original sample instead of a bootstrap replica will reduce bias. On the other hand, choosing randomly the split point of each node will reduce variance.

In terms of computational cost, and therefore execution time, **the Extra Trees algorithm is faster**. This algorithm saves time because the whole procedure is the same, but it randomly chooses the split point and does not calculate the optimal one.

From these reasons comes the name of Extra Trees (Extremely Randomized Trees)

Comparison

In this section, we are going to compare the performance of the two algorithms. For that purpose, we are going to analyze the obtained results when using different datasets.

In order to generate these datasets, the function *make_classification* of the *sklearn* python library has been used. It has been chosen to use synthetic datasets instead of real ones because, in this way, the properties of the dataset can be tuned. Furthermore, this function allows us to select the number of features and their characteristics.

Therefore, this gives us the opportunity of analyzing the performance of the two algorithms in a classification problem, using datasets with different properties. In total, five datasets have been generated in which all of them have 1000 samples, 20 features, and 2 classes (binary classification). Moreover, 30% of the samples are assigned randomly. In this way, introducing noise make the classification harder. Moreover, the chosen differences between the features of the datasets are the following:

	Informative	Repeated	Redundant	Random
Dataset 1	20	0	0	0
Dataset 2	2	0	0	18
Dataset 3	2	5	5	8
Dataset 4	2	0	18	0
Dataset 5	2	18	0	0

So, we can analyze if the two algorithms behave differently when having redundant, noisy, and repeated features.

Besides, the sklearn implementation of the two algorithms has been used. In addition, as for the parameters, the same number of estimators has been chosen, 100, leaving the rest of the parameters by default. Furthermore, **cross-validation** has been applied with K=3.

Results

The following table summarizes the obtained results comparing the F-Score and the execution time:

	Extra Trees		Random Forest	
	F-Score	Time(s)	F-Score	Time(s)
Dataset 1	0.773 ± 0.009	0.213	0.778 ± 0.029	0.345
Dataset 2	0.775 ± 0.027	0.206	0.784 ± 0.030	0.349
Dataset 3	0.807 ± 0.015	0.206	0.811 ± 0.019	0.291
Dataset 4	0.755 ± 0.016	0.238	0.788 ± 0.023	0.283
Dataset 5	0.758 ± 0.017	0.182	0.782 ± 0.022	0.357

From the above results, we can see that results barely vary for the two algorithms. It seems that the Random Forest gets better results, but the difference is very small. As mentioned above, the variance of the Extra Trees is lower, but again, the difference is practically insignificant. It is remarkable that results vary more for datasets 4 and 5 where most of the features are redundant or repeated.

Moreover, as expected, **Extra Trees is much faster**. This is because instead of looking for the optimal split at each node it does it randomly.

Conclusions

In this post, we have studied the difference between Extra Trees and Random Forest algorithms, comparing the obtained results when applying them for different datasets.

In the comparison that has been made, the obtained results are practically the same for both algorithms. It is worth noting the difference in the execution time, where the Extra Trees is much faster. Therefore, if you have any doubt about which of the two ensembles to use, it seems a good idea to use Extra Trees since the same result is obtained in a faster way.

References

[1] Geurts, Pierre, Damien Ernst, and Louis Wehenkel. “Extremely randomized trees.” Machine learning 63.1 (2006): 3-42.