Exoplanets detection using auto-tuned random forest

Lorenzo Loconte

January 8, 2020

Abstract

This work consists of identifying exoplanets using random forest which hyperparameters are automatically optimized. The model is trained using the processed data that comes from NASA's Kepler project of discovering exoplanets (e.g. planets outside our solar system). The hyperparameters of the model are optimized and cross-validated with random search, a simple yet effective scalable method for hyperparameters optimization. The data used in this work can be found at **https://exoplanetarchive.ipac.caltech.edu**.

1 Introduction

Kepler Object of Interest The dataset used to train the model is composed by a list of features regarding exoplanets. Each tuple have a label indicating if the corresponding exoplanet is candidate, false positive or confirmed. The selected features for the classification task are the following:

1. Orbital Period [days]
2. Impact Parameter
3. Transit Duration [hrs]
4. Transit Depth [ppm]
5. Planet-Star Radius Ratio
6. Fitted Stellar Density [g/cm**3]
7. Planetary Radius [Earth radii]
8. Orbit Semi-Major Axis [AU]
9. Inclination [deg]
10. Equilibrium Temperature [K]
11. Insolation Flux [Earth flux]
12. Stellar Effective Temperature [K]
13. Stellar Surface Gravity [log10(cm/s**2)]
14. Stellar Radius [Solar radii]
15. Stellar Mass [Solar mass]
16. RA [decimal degrees]
17. Dec [decimal degrees]
18. Kepler-band [mag]

The tuples corresponding to features of candidate exoplanets are discarded. Only confirmed and false positive ones are included in the train and the test sets in order to reduce noises and improve the learning process. The preprocess method applied to the dataset is the standard normalization.

Random Forests and Hyperparameters The model used for the classification task is a random forest, an ensemble of random decision trees. The prediction of the random forest is computed as the mode of each tree's classification. Each tree of the forest is trained on a subset of the features and samples. The hyperparameters of the random forest are optimized and cross-validated using random search. The following hyperparameters are optimized:

- Number of estimators (e.g. the number of trees)
- The minimum number of samples required to split an internal node
- The percentage of features to use for each tree

Model	Settings	Precision	Recall	F_1
k-NN	k=5, Euclidean norm, Uniform weights	0.922	0.829	0.873
SVC	Regularizer=1.0, RBF kernel, $\gamma = (m\sigma^2)^{-1}$	0.925	0.875	0.899
2-layer NN	100 hidden units, ReLU activation, Adam	0.930	0.917	0.926
Random Forest	**Auto-tuned Hyperparameters**	**0.943**	**0.939**	**0.941**

Table 3: Precision, recall and F_1 metrics of several models for comparison. The hyperparameters of *Random Forest* were optimized automatically. In this table it was built with the following hyperparameters: 90 estimators, 4 minimum samples to split an internal node and 5 features per tree.

2 Hyperparameters Tuning

The hyperparameters optimization task consists to find the hyperparameters for a model that maximize a certain score. In this work we refer to hyperparameters as the ones that cannot be trained (e.g. the ones described in the previous section). There are a lot of different techniques for hyperaparameters optimization, some of them are the following:

- Random Search
- Grid Search
- Hyperbands
- Genetic methods
- Bayesian optimization

In this work, given the restricted hyperparameters search space, we used random search as the hyperparameters algorithm. It's a simple method that consists of random sampling some points in the hyperparameters space and cross-validating them on the validation set. After some iterations we pick the best hyperparameters point corresponding to the best model. Even if it is a simple algorithm, it is well scalable on multiple CPUs because every point in the hyperparameters space is independent from the others. In this work the hyperparameters search space can be defined formally as:

$$\Omega = \mathbb{N}^* \times \mathbb{N}^* \times (0, 1]$$

Table 1 shows the subset of the search space used in this work.

Hyperparameters	Values
Estimators number	$\{10, 20, ..., 100\}$
Minimum samples to split	$\{2, 4, 8, 16, 32\}$
Features percentage	$\{0.1, 0.2, ..., 1.0\}$

Table 1: Hyperparameters space subset

The optimization task consists to find a point in the hyperparameters space:

$$\omega = (n, k, p) \in \Omega$$

where n is the number of estimators, k is the minimum samples to split, p is the features percentage for each tree, such that the random forest built from these hyperparameters maximize a certain score. The choice of this score is dependent from the task. For simplicity we use the F_1 score (e.g. we want to maximize both *Precision* and *Recall*).

3 Conclusion and Results

As you can see from Table 2 the false positives count (the number of examples, which are predicted as exoplanets, that are not) and the false negatives count (the number of actual exoplanets not being discovered), are pretty low.

		Actual	
		Positive	Negative
Predicted	Positive	**408**	52
	Negative	56	**861**

Table 2: Confusion matrix over the test set.

As you can see from Table 3 the auto-tuned random forest used in this work obtained a way better precision and recall. For this task we want to have a better recall because we want to maximize the number of actual exoplanets discovered. In the end we present Table 4 that shows the importances of the first five out eighteen features used.

Feature #	Description	Importance
7	Planetary Radius	0.137
5	Planet-Star Radius Ratio	0.119
8	Orbit Semi-Major Axis	0.083
1	Orbital Period	0.069
6	Fitted Stellar Density	0.066

Table 4: Features importances in descending order.