Hyperparameter Optimization

Milana M. Wolff

April 09, 2023

1 Introduction

In this assignment, we optimize relevant hyperparameters for a small selection of classification models using the wine quality dataset. This widely used dataset contains a variety of physicochemical input features, such as wine density and acidity, along with expert ratings for red Vinho Verde wines. We approach the hyperparameter optimization problem by selecting three classifiers with competitive performance under default hyperparameter configurations: Ridge, bagging, and random forest. We then conduct hyperparameter optimization with grid search cross-validation and nested resampling (3 outer folds, 10 inner folds per hyperparameter configuration).

2 Dataset Description

The dataset used for this assignment contains physicochemical quantitative input features and sensory quantitative output features (i.e., an expert wine score) for the red variant of the Portuguese "Vinho Verde" wine [1]. The dataset includes 1599 observations and eleven input features, including fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content. According to the UC Irvine Machine Learning Repository website, "the classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones)", with a total of 1319 observations rated as 5 or 6 and a mere 28 observations rated with the highest and lowest scores (3 and 8) [2]. This robust dataset includes no missing values to be imputed. We use the eleven listed input features to predict the wine quality measurement as the target during hyperparameter optimization.

3 Experimental Setup

We use Python 3.10.12 (GCC 11.4.0) in a Jupyter/interactive Python notebook on Google Colaboratory, as well as a functionally identical pure Python file running on the Beartooth/Teton cluster (default Python version 3.7.16 (GCC 11.2.0). After importing the `scikit-learn` package, we use Ridge, bagging, and random forest classifier models. We optimize the alpha, tolerance, solver, and maximum iteration hyperparameters for the Ridge classifier models; the number of samples, maximum estimators, and maximum feature hyperparameters for the bagging classifier models; and the number of estimators, splitting criterion, and maximum tree depth for the random forest classifier models.

See the table documenting the different hyperparameter combinations for each model below.

Ridge	Alpha: [1.0, 1.1, 2.0, 5.0]	Tol: [0.0001, 0.001, 0.01, 0.1]	Solver: ["svd", "cholesky", "lsqr" "sparse_cg"]	Max_Iter: [100, 200, 500, 1000, 10000, None]
Bagging	N_Estimators: [100, 500, 1000, 10000]	Max_Samples: [0.1, 1.0, 2, 5]	Max_Features: [0.1, 1.0, 2, 5]	
Random Forest	N_Estimators: [100, 500, 1000, 10000]	Criterion: ["gini", "entropy", "log_loss"]	Max_Depth: [None, 2, 3, 5, 10]	

We use a grid search approach to accommodate different parameter types, such as solver for the Ridge models and criterion for the random forest models. While ease of implementation is a significant consideration, note that `scikit-learn` supports implementation of both grid and random hyperparameter searches. Furthermore, grid search enables parallelization, which enables scaling of the code if needed. While grid search has disadvantages, such as low resolution, combinatorial explosion inefficiencies, and may result in irrelevant searches in parameter space, this approach seemed suitable for a first attempt at HPO.

We load the wine data using the pandas library and use all eleven features for classification without additional pre-processing steps. For each machine learning algorithm, we use a nested resampling approach. We use a train/test split of 80/20. Using the outer training data, we perform nested resampling by performing 10-fold cross validation on each hyperparameter configuration using `cross_validate` method from `scikit-learn`. For each of these inner 10 folds, we use an 80/20 train/test split of the outer training set to tune model parameters for a given hyperparameter configuration. We measure the performance best model parameters for a given hyperparameter configuration using balanced accuracy as the scoring metric and report the performance of the best model for each of the outer folds.

4 Results

Using a for-loop approach, performing hyperparameter optimization on the Ridge classifier yields a mean balanced accuracy of 0.2346 with alpha = 1.0, tol = 0.01, the `sparse_cg` solver, and 100 iterations maximum. On the bagging classifier, HPO yields a mean balanced accuracy of 0.3258 with 500 estimators, 1.0 as the `max_sample` value, and 5 features maximum. On the random forest classifier, a mean balanced accuracy of 0.3191 is achieved with 100 estimators, the `entropy` criterion, and a maximum tree depth of 10.

Using the functionally identical (albeit faster) built-in grid search approach for the Ridge classifier yields a balanced accuracy score of 0.230 $\pm$ 0.027 with the following hyperparameter values: {'alpha': 1.0, 'max_iter': 100, 'solver': 'svd', 'tol': 0.0001}. Based on the different results between approaches, we can conclude that the tolerance and solver hyperparameters matter less than the `alpha` and `max_iter` values–which are set to the classifier defaults in both hyperparameter results we obtain.

The same approach for the bagging classifier yields a generalized balance accuracy score of 0.267 $\pm$ 0.031 with the following hyperparameter values: {'max_features': 1.0, 'max_samples': 0.1, 'n_estimators': 1000}. Note that the number of estimators is far greater than the default hyperparameter value (10) and that max_samples differs from the default value (1.0), indicating that the classifier only samples a tenth of the provided training set to train each estimator.

The grid search method yielded a generalized balanced accuracy score of 0.265 $\pm$ 0.032 for the random forest classifier with the following hyperparameters: `criterion: 'entropy', 'max_depth': 5, 'n_estimators': 100}`.

The Bayesian optimization approach yielded a generalized balanced accuracy score of 0.234 $\pm$ 0.023 for the Ridge classifier with the following hyperparameter values: `alpha=1.1774681965430613, max_iter=97063, solver='svd', tol=0.00111048408666268`. The Bayesian optimization process is still running on Beartooth and Colaboratory, results pending.

5 Code

`https://github.com/COSC5557/hyperparameter-optimization-mwolff2021`

References

[1] In: (). URL: `http://www.vinhoverde.pt/en/`.

[2] Paulo Cortez, A. Cerdeira, F. Almeida, et al. "Wine Quality". In: (2009). DOI: https://doi.org/10.24432/C56S3T.