Hyperparameter Optimization

Milana M. Wolff

May 04, 2024

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. In addition, we use a functionally identical pure

Classifier	Ridge	Bagging	Random Forest
Numerical Hyperparameter Ranges	alpha: (1.0, 5.0) tol: (0.0001, 1.0) max_iter: (100, 100000)	n_estimators: (100, 10000) max_samples: (0.01, 1.0) max_features: (0.01, 1.0)	n_estimators: (100, 10000) max_depth: (2, 10)
Numerical Hyperparameter Priors	log-uniform (all)	log-uniform (all)	log-uniform (all)
Categorical Hyperparameters	'svd', 'cholesky', 'lsqr', 'sparse_cg'		'gini', 'entropy', 'log_loss'

Python file running on the Beartooth/Teton cluster (default Python version 3.7.16 (GCC 11.2.0)), with an allocation of 32 GB of memory and 5 hours 10 minutes runtime, as two of the classifiers do not complete hyperparameter optimization when executed on Colaboratory. 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. These hyperparameters control regularization strength, the precision of the solution, the solver used in computational routines to obtain the best solution, and the maximum number of iterations for conjugate gradient solver, respectively [3]. For the bagging classifier, we use the maximum samples (as a proportion of total samples), number of estimators, and maximum feature hyperparameters [4]. We use the number of estimators, splitting criterion, and maximum tree depth for the random forest classifier models [5].
The ranges and priors for numerical hyperparameters and the options for categorical hyperparameters are tabulated below.

We use a Bayesian search approach with the `BayesSearchCV()` method provided by `scikit-learn`. 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 listed above, we use a nested resampling approach with a three-fold outer cross-validation and ten-fold inner cross validation. We use balanced accuracy as a scoring criterion in the Bayesian search to account for the underlying class imbalance problem in the wine quality dataset.

4 Results

For each outer loop of the cross-validation, we report the average results of the inner 10-fold cross validation, the generalization score, the balanced accuracy

Classifier	Ridge	Bagging	Random Forest
Generalization Score	0.236 ± 0.022	0.229 ± 0.022	0.331 ± 0.045
Score on Left Out Data (inner CV)	0.2318	0.2245	0.2963
Score on Test Data (outer CV)	0.2322	0.2313	0.3184
Generalization Score (Default Hyperparameter Settings)	0.247 ± 0.027	0.340 ± 0.042	0.351 ± 0.043

on the outer CV test dataset, and the best score on the left-out data for the inner CV yielding the best estimator. These results are tabulated for the Ridge, bagging, and random forest classifiers below.

The hyperparameter configuration for the best estimator for the Ridge classifier in the search space defined was: `alpha`=1.267774542235981, `max_iter`=22377, solver='cholesky' and `tol`=0.002318281928932147. Notably, the `tol` hyperparameter has no effect if the Cholesky solver is used.
In comparison, the default hyperparameters for the Ridge classifier are `alpha`=1.0, the default maximum number of iterations is determined by scipy.sparse.linalg., the solver is automatically chosen to fit the data, and `tol`= 0.0001. For the bagging classifier, the best hyperparameter configuration obtained used `max_features` = 0.0329 (the maximum number of features used for classification, expressed as a fraction of the total number of features), `max_samples` = 0.9074, and `n_estimators` = 6560.
In comparison, the default hyperparameter settings are `max_features` = 1.0 (i.e., all features), `max_samples` = 1.0 (all samples), and `n_estimators` = 10. The default estimator is a decision tree, as the bagging classifier is an ensemble model.
The random forest classifier with the best performance in nested resampling used `max_depth` = 10, `criterion` = 'gini', and `n_estimators` = 6560 as the hyperparameter configuration.
The default hyperparameter settings are `max_depth` = None (i.e., no limit on maximum tree depth), `criterion` = 'gini', and `n_estimators` = 100.
Compared to default hyperparameter settings, the optimized hyperparameter settings did not improve performance, and yielded slightly worse results in all cases. This may be the result of a low number of iterations of the Bayesian search CV (due to memory and time constraints) and local minima or, in the specific case of the random forest classifier, limiting the maximum tree depth of the search space too greatly to achieve optimal performance. In all other cases, the default hyperparameter settings were configurations possible within the Bayesian search space.

5 Code

https://github.com/COSC5557/hyperparameter-optimization-mwolff2021-1

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.

[3] URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html#sklearn.linear_model.RidgeClassifier.

[4] URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html#sklearn.ensemble.BaggingClassifier.

[5] URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.