Wildcard Project – Surrogate Optimization & Space Interpretation

COSC5557 – Practical Machine Learning

William Baumchen

12/13/2023

1 - Introduction

When considering the optimization of a pipeline, investigation of the search space and testing of different optimization methods can be quite valuable. In this report, the further optimization of a relatively simple pipeline will be conducted on the white wine quality dataset, utilizing surrogate optimization as a speedup and comparison with previous attempts, and the resulting optimization evaluations considered in order to better interpret the search space.

2 – Dataset Description

The dataset used is the white wine quality dataset, containing 4,898 observations, with 11 features, all of which are real numbers taken from a continuous range. There is one 'target' feature, with seven possible classes. There are no missing values in the observations and target feature.

3 – Experimental Setup

In this exercise all computation was done utilizing MATLAB, more specifically the Statistics and Machine Learning Toolbox. Firstly, the dataset was split for evaluation; a fifth of the original dataset was set aside for testing, and the remaining 80% of the data was kept for model training. The pipeline was constructed in three steps – a feature normalization step, a feature reduction step utilizing principal component analysis, and a hyperparameter optimization step conducted using three different classification machine learning models. The optimization searched through the hyperparameter space, which included whether to normalize the data, whether or not to conduct feature reduction, and to what degree, and finally what model to use and which hyperparameters to select for that model. The pipeline optimization objective was the validation classification loss computed during the optimization.

The third component of the pipeline was the selection of the machine learning model and the hyperparameters for that model. This pipeline considered three different classification machine learning models. These models were a binary decision tree classifier, an ensemble classifier, and a k nearest neighbors classifier. For the binary decision tree, the optimization considered the minimum leaf size. The range of parameters used in the hyperparameter optimization was based on the default options set by the fitctree function in MATLAB and reported in Table 1.

Table 1 – Decision Tree Hyperparameter Ranges

Hyperparameter Optimization Ranges – Decision Tree	
MaxNumSplits	Log-scaled integers in [1,max(2,NumObservations-1)]

For the ensemble classifier, the optimization considered the method used by the classifier. The range of parameters used in the hyperparameter optimization was based on the default options set by the fitcensemble function in MATLAB and reported in Table 2.

Table 2 – Ensemble Hyperparameter Ranges

Hyperparameter Optimization Ranges – Support Vector Machine	
Method	Searches between 'Bag','AdaBoostM2','RUSBoost'

For the k-nearest neighbor model, the optimization considered the distance metric, choosing among several predefined metrics, the number of nearest neighbors in the dataset to find for classifying each point when predicting, and whether or not to standardize the training data. The range of parameters used in the hyperparameter optimization was based on the default options set by the fitcknn function in MATLAB and reported in Table 3.

Table 3 – KNN Hyperparameter Ranges

Hyperparameter Optimization Ranges – K-Nearest Neighbor	
Distance	Searches among 'cityblock', 'chebychev', 'correlation', 'cosine', 'euclidean', 'hamming', 'jaccard', 'mahalanobis', 'minkowski', 'seuclidean', and 'spearman'
NumNeighbors	Searches among positive integer values, by default log-scaled in the range [1, max(2,round(NumObservations/2))]
Standardize	Searches among ‘true’ and ‘false’

For each of models used, k-fold cross validation with 5 folds was used to avoid overfitting and bias. These folds were repartitioned with every individual model fitting. In this exercise, a surrogate optimization was carried out utilizing the MATLAB function surrogateopt [1]. Here, a total of 400 iterations were given. Once the pipeline optimization was complete, a final evaluation of the model’s validation loss was carried out on the outer set of testing data, and the resulting data was compared to a Bayesian optimization carried out in a previous exercise. Both optimizations used the same seed for random generation. In addition, parallel coordinate plots were constructed from the evaluation results found during the Bayesian optimization, an evaluation for each iteration of the optimization, to better understand the search space.

4 – Results

The pipeline optimization algorithm above was run according to the given conditions. Fig. 1 below shows the entirety of the optimization results, where the results are the MATLAB classification loss function for each of the optimization evaluation points. In this exercise, the pipeline arrangement with the best performance was a non-normalized, classification ensemble model using PCA, with 9 features retained, and utilizing the 'Bag' Method.

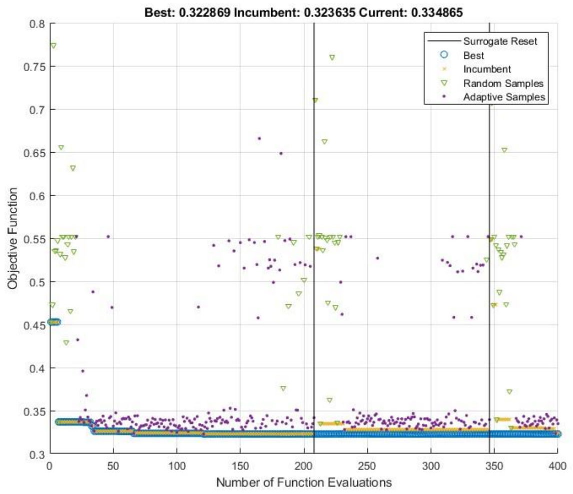

Figure 1 – Bayesian optimization over iterations

That is, the hyperparameters found as 'optimal' using the surrogate optimization were solver: 1 (1 being an ensemble model), featureNum: 9 (9 meaning pca occurred, and 9 features were retained), normVal: 0 (no normalization was used), and Method: 'Bag' (the learning method the ensemble model used was the Bag method). This corresponds almost exactly to the parameters found using the previous Bayesian optimization, and took 121 iterations, or approximately 618.65 seconds. That is, the Bayesian optimization utilized pca, but did not remove any features, while the surrogate optimization used pca but only used 9 of the newly found features.

The optimized ensemble had a minimum observed evaluation average cross-validation loss of 0.32287, and an average cross-validated loss on the test data of 0.37807. The optimized ensemble had a minimum observed evaluation average cross-validation loss of, and an average cross-validated loss on the test data of. This is in comparison with the results from the Bayesian optimization, which had an average cross-validation loss of 0.32177 and an average cross-validated loss on the test data of 0.37722. The Bayesian optimization performed better in finding an optimal model. However, the Bayesian optimization took 145 iterations to reach that point, with a time elapsed of 378.0485 seconds. As such, using surrogate optimization in this case is less useful than performing Bayesian optimization, as it took a bit under twice the time the Bayesian optimization did to find the 'optimal' hyperparameter configuration for the pipeline.

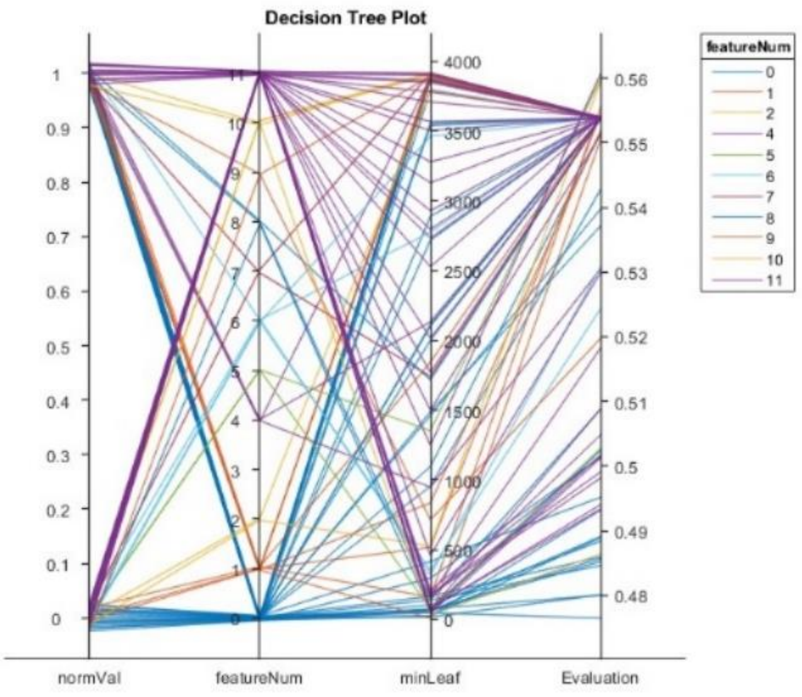

Figure 2 – Parallel Coordinate Plot, Decision Tree

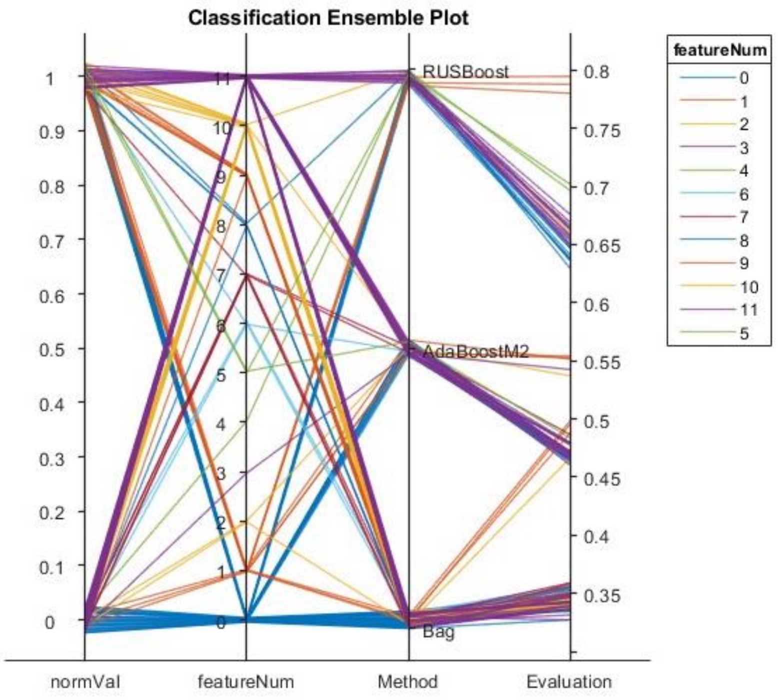

Figure 3 - Parallel Coordinate Plot, Ensemble

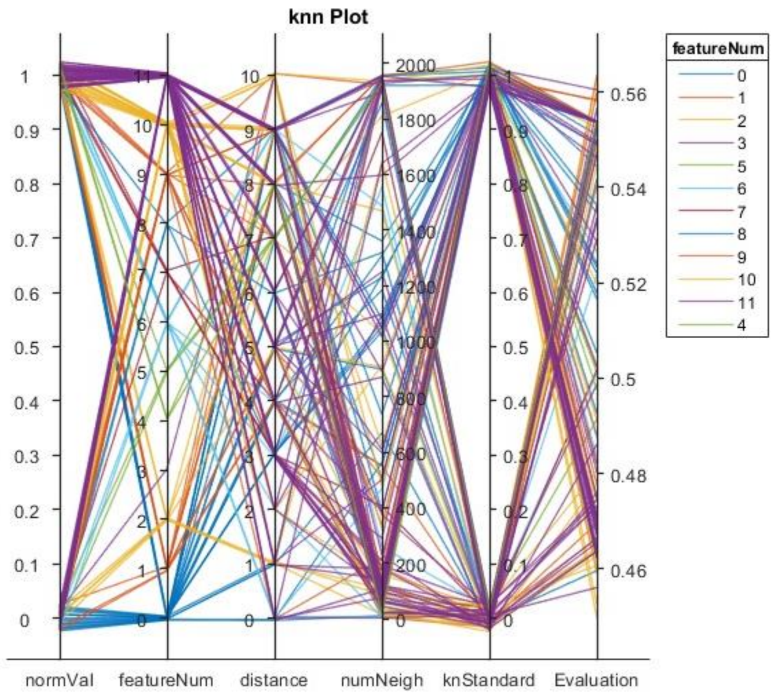

Figure 4 – Parallel Coordinate Plot, KNN

Above are the three different parallel coordinate plots of the three different models, found using the evaluations from the optimization. That is, in the Bayesian optimization an evaluation on the pipeline for some hyperparameter configuration is carried out at each iteration. Each of these configurations, and the resulting metric, are sorted by the type of model used and plotted above. Here, the Evaluation metric is the cross-validated classification loss of the model for that iteration trained on the training dataset.

A simple examination reveals that the ensemble model was the clear winner in terms of the cross-validated loss. As is easily visible in Fig. 3, the 'Bag' Method was the best choice for that hyperparameter, easily outperforming the other two choices. In addition, when performing feature reduction though pca, the best performing results came from either not utilizing feature reduction, or using it, and retaining many the newly found features. Finally, observing the plot, the normalization of the data did not seem to significantly impact the result, with a roughly even split in the well-performing models either using it or not.

References

[1] The MathWorks, Inc. (n.d.). surrogateopt. Surrogate optimization for global minimization of time-consuming objective functions - MATLAB. https://www.mathworks.com/help/gads/surrogateopt.html