Hyperparameter Optimization

Finn Tomasula Martin

COSC-4557

1 Introduction

Most machine learning models aim to find parameter values that best fit the given data to a predictor. These models are usually quite good at finding the optimal parameter values which means the user does not have to do much work in the model building process. But, these models often have parameters of their own called hyperparameters that must be set before the model is run and often greatly affect the success of the model. So, it is important to find the best hyperparameter values before building a final model. There are many methods used for hyperparameter optimization but in this report we will discuss how to use Bayesian hyperparameter optimization on a couple ML algorithms to determine the best algorithm and hyperparameter configuration for a given problem.

2 Data

The dataset we will be using for this exercise is called winequality-red.csv. This dataset contains 1599 observations on different red wines. The target variable we are trying to predict is overall wine quality based on 11 features. The quality variable is an ordinal value ranging from 3 to 8 where 3 is the lowest quality and 8 is the highest. To simplify our work we will create a new target that classifies wine as good or bad based on if it's quality is above the mean or not.

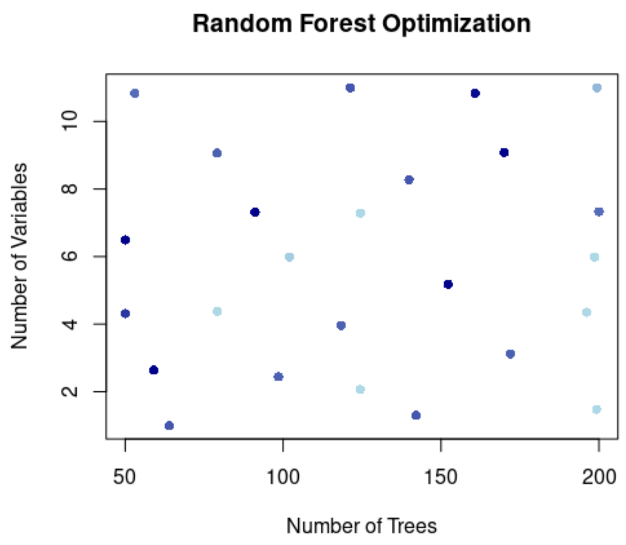
3 Experimental Setup

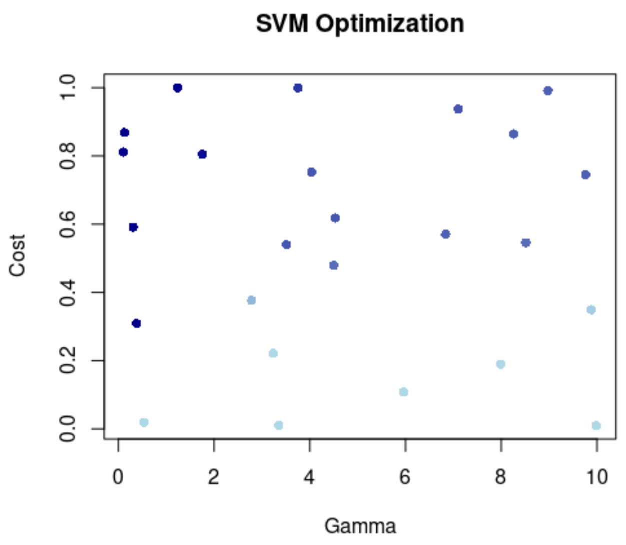
For this analysis we will be using R and the following libraries: caret, randomForest, e1071, and rBayesianOptimization. We will be taking a look at two different machine learning algorithms and optimizing two hyperparameters each. These algorithms and their hyperparameters are random forest with respect to number of trees and number of variables, and SVM with respect to gamma and cost. As mentioned earlier the optimization technique we will use is Bayesian HPO. The evaluation we will be using is average classification accuracy with 10-fold cross validation. The function we will be using to optimize our parameters is called BayesianOptimization which is part of the rBayesianOptimization library. For each of our chosen algorithms we will need to define the following in order to run the optimization on it: objective function, parameter bounds, number of initial points, number of iterations, acquisition function and epsilon. In order to keep our analysis consistent we will use the same values for number of initial points, number of iterations, the acquisition function and epsilon for both algorithms. The number of initial points defines the initial random parameter configurations we will start with. We will choose 5. The number of iterations defines the number of times the function will choose a new configuration and evaluate. We will choose 20. The acquisition function defines the type of function to be used when picking the next parameter configuration. We will be using probability of improvement. Finally epsilon is a weight added to the process intended to balance exploration vs exploitation. We will choose 0.1 in order to favor exploration and ideally escape the local optima. The objective function and the parameter bounds will need to be different between the two algorithms since they will be using a different model with different parameters. The objective function defines the model and evaluation of each configuration that is chosen. The parameter bounds define the bounds that values for each parameter will be chosen from. For random forest we will bound the number of trees between 50 and 200 and the number of variables between 1 and 11. For SVM, we will bound gamma to values between 0.1 and 10, and cost between 0.01 and 1. The random forest objective function does the following: First it floors the chosen values for the parameters since both number of trees and number of variables are integer values. Then we will run the random forest model on each of the 10 folds of the data with the chosen parameters and calculate the classification accuracy of each. Finally, we compute the average accuracy and return that as the score. The SVM objective function is very similar with the SVM model in place of the random forest model. We do not need to floor the parameter values since gamma and cost do not need to be integers. Then we run the SVM model on each of the 10 folds and compute the classification accuracy of each. Then, we return the average accuracy. Once both optimizations have been run, we will have the optimal hyperparameter configuration for each one and can pick the best one.

---

4 Results

When we run random forest without any tuning we get an average classification accuracy of 0.821. When we run SVM without any tuning we get an average classification accuracy of 0.764. After tuning we were able to get random forest up to 0.832 and SVM up to 0.772. From here we can choose the optimized random forest configuration since it yielded the highest average classification accuracy. The configuration that led to this result is number of trees: 98 and number of variables: 2. Additionally here are plots for the optimization so we can observe how the process worked (darker color indicates a better evaluation):





Overall we were able to increase the predictive power of our model by about a percentage. We may be able to increase these results with any number of factors including number of iterations and possible value ranges. Additionally, this particular problem may not need very much parameter optimization to be effective.

---

5 Code

See HPO.R for all code.

https://github.com/COSC5557/hyperparameter-optimization-ftomasul/blob/main/HPO.R

Sources

Bayesian Optimization in R:

https://cran.r-project.org/web/packages/rBayesianOptimization/rBayesianOptimization.pdf

Chat GPT 3:

What are suitable value ranges for gamma and cost when optimizing an SVM model?