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 three hyperparameters each. These algorithms and their hyperparameters are random forest with respect to number of trees, number of variables and node size, and SVM with respect to kernel, gamma and cost. As mentioned earlier the optimization technique we will use is Bayesian HPO through the BayesianOptimization function which is part of the rBayesianOptimization library. In order to evaluate our optimization we will have to use nested optimization in order to ensure an unbiased

performance estimate. To do this we will split our data into a training set (80% of the data) and a testing set (20% of the data). This will act as our outer evaluation where we can compare our base algorithms to our optimized algorithms using classification accuracy as our evaluator. While optimizing we will run 10-fold cross validation on our training set and compute average classification accuracy between the folds. This will act as our inner evaluation and will be used by our optimization process to evaluate a given set of hyperparameters. The optimization process will be the following: First for each algorithm we need to define an objective function. This function will take as arguments a hyperparameter configuration. It will then run a model and evaluate the results, using the inner evaluation process described above. This function will have to be different for random forest and SVM since they will be running different models with different parameters. Next we will need to define the bounds of the hyperparameters we are trying to optimize. For random forest we will define the following bounds, number of trees: 50 – 500, number of variables: 1 – 11, node size: 1 -10. For SVM we will define the following bounds, kernel: linear, polynomial, radial or sigmoid, gamma: 0.1 – 1, cost 0.1 – 1. Then we have to make several choices about our Bayesian process. First will be the number of initial points that the process starts from. We will choose 5. The value of 5 was chosen more or less randomly as it seems to offer a decent starting point for the process but more or less initial points could be more optimal. Second is the number of iterations that will be run. We will choose 100. The value of 100 was chosen because when the process was run with less iterations, such as 20 or 50 the optimized configuration did not consistently improve the evaluation, but it did with 100. Next is the acquisition function used for picking the next set of hyperparameters. We will choose probability of improvement. Probability of improvement was chosen over expected improvement and upper confidence bound because it yields similar results with much less computation time for this problem. When the optimization was run with expected improvement or upper confidence bound, iterations could take up to 4 seconds to complete while interations with probability of improvement only took up to half a second at most. Finally, we will need to choose an epsilon value in order to balance exploration and exploitation. We will choose 0.01. When higher epsilon values were chosen such as 0.1 meaning we are favoring exploration, the final optimized configuration did not yield any improvement. When lower values of epsilon were chose such as 0 which means we are favoring exploitation, the final optimized configuration did yield improvement. This tells us that for this problem we should favor exploitation, so we set our epsilon at 0.01 in order to do just that but we also want to be able to escape the local optima which is why we don’t set it to 0. Using all this information we can can run the Bayesian optimization on each algorithm. For each algorithm it will choose 5 random hyperparameter configurations and evaluate them using the objective function. Then it will use the acquistition function to pick a new configuration and evaluate

that. It will repeat this process 100 times and once it is finished it will give us the best hyperparameter configuration that it found. Then we can run our algorithms with default hyperparameters and then again with our optimal parameters to see if we had any improvement.

---

4 Results

Running our base models with default parameters, random forest yielded a classification accuracy of 0.803 and SVM yielded a classification accuracy of 0.762. After running our optimization process we are able to get the following optimal configurations: random forest where number of trees is 428, number of variables is 1 and node size is 1, which yielded a classification accuracy of 0.821; and SVM where the kernel is radial, gamma is 0.436 and cost is 1, which yielded a classification accuracy of 0.774. Since the optimized random forest model yielded the best overall evaluation we will choose that one as our final model. Overall we were able to get an x percentage increase in our classification accuracy by optimizing hyperparameters.

---

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?