AutoML Assignment 1

Evan Meltz, Giulia Rivetti (Group 36)

1 Introduction

In this assignment we demonstrate the working of the SMBO algorithm and how it can be used to optimize the hyperparameters of two different scikit-learn algorithms, namely Support Vector Machines as well as Adaboost using three distinct datasets from OpenML. We then plotted our results and compared them against grid and random search, to understand which method can find the optimal hyperparameters which will make the best model performance for the various datasets.

2 SMBO optimizing SVM and Adaboost

Sequential Model Based Optimization (SMBO) or Bayesian Optimization is an optimization technique used to tune hyperparameters in order to find the best hyperparameter configurations for a certain model. In particular, to determine if a configuration is good for a model or not, SMBO uses an acquisition function. For this purpose, the expected improvement was calculated for each configuration, and the best set of hyperparameters were selected by looking at the highest value of the expected improvement. Once a configuration has been selected, it is added to a run list so that the model can be trained on it in the next iteration.
For the SMBO optimizing SVM and Adaboost algorithms, first, we have defined a configuration space for the hyperparameters (gamma and C) of the SVM classifier. For the lower and upper bounds of the two hyperparameters we chose to maintain the ones that were given in the 'test.py' file, because those are the configurations which gave us the best results. We have also tried to restrict and enlarge the interval between lower and upper bound of both gamma and C, but in this case we have obtained good performance only for one or two dataset, but not for all the three of them. For this reason we decided to maintain the configuration space unaltered.
Using the SMBO class given, the SVM algorithm was optimized by sampling 10 initial configurations obtained by tuning gamma and C hyperparameters. The obtained configurations were then used to initialize the SMBO model. Next, we iterated through a loop 16 times. At each iteration of the loop we fit the Gaussian Process model on the current run list. Then, we sampled 64 new configurations randomly generated from the configuration space, that we previously defined, and then we select the best configuration. At this point:

- Regarding the SVM algorithm, we initialized an SVM classifier setting as its hyperparameters the values for gamma and C that we have generated before. We employed the Radial Basis Function kernel for the classifier
- Concerning Adaboost, we have also initialized an SVM classifier, which, in this case, we have used as the base-estimator for the Adaboost classifier ensemble.

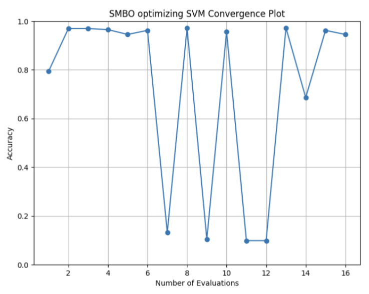
In both cases (SVM and Adaboost) we have then trained the classifier based on the training data and we compute the performance (in particular the accuracy score) of the trained classifier on a validation set. Finally we updated the run list with the best configuration obtained at this iteration.

3 Datasets

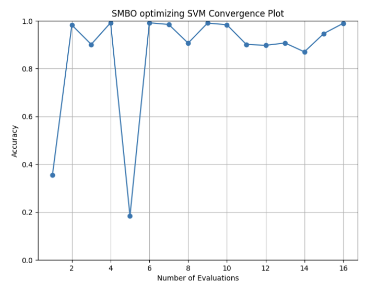
From OpenML we selected three datasets: $mfeat-karhunen$ (id:16), har (id:1478) and $semeion$ (id:1501), with dimensions: (2000, 64), (10299, 561) and (1593, 256) respectively. As it can be noticed, we chose datasets of different dimensions and complexity, in order to see how the different optimization techniques that we have employed are affected by different datasets. Moreover, those three datasets present a different level of skewness, with $mfeat-karhunen$ and har being significantly right-skewed.

4 Experiment results

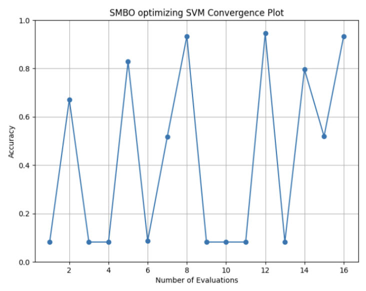
Considering the performance of SMBO optimizing SVM, we can observe that the algorithm does not seem to be affected by neither the dimensions of the dataset, nor by its skewness; indeed it results in high accuracies for the three datasets (larger than 0.94) – see *Figure* 1. In fact, in the original SVM implementation, the complexity of the algorithm's training is highly dependent on the size of the dataset; however, here the results do not mirror that, because we have not employed the classical SVM algorithm by itself, but rather we have optimized it through SMBO.



(a) Dataset 'mfeat-karhunen'



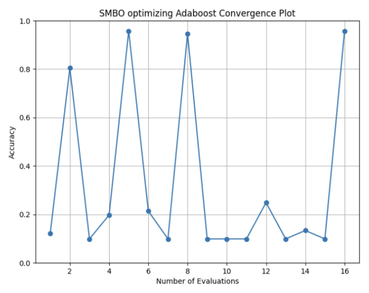
(b) Dataset 'har'



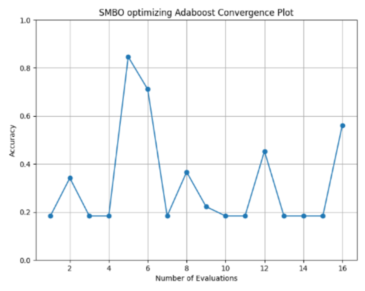
(c) Dataset 'semeion'

Figure 1: Plots of accuracy versus the number of iterations for the three datasets when employing SMBO to optimize SVM.

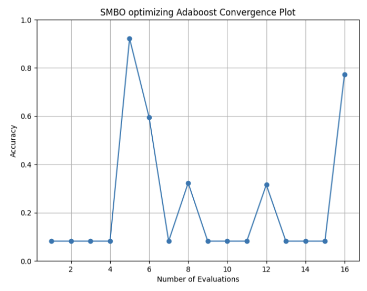
Considering SMBO optimizing Adaboost, we can notice that the algorithm reaches good accuracies (higher than 0.94) with smaller datasets, namely $mfeat-karhunen$ and *semeion*. However, the accuracy measured on the *har* dataset is only 0.85, showing that Adaboost optimization is actually affected by the dimensions of the datasets. In general, we have obtained better results with SMBO optimizing SVM with respect to the Adaboost optimization, which is also much slower. Moreover, we have noticed that both optimization's training time are affected by the dimensions of the datasets; indeed SMBO requires a higher computational cost for larger datasets.



(a) Dataset 'mfeat-karhunen'



(b) Dataset 'har'



(c) Dataset 'semeion'

Figure 2: Plots of accuracy versus the number of iterations for the three datasets when employing SMBO to optimize Adaboost.

5 Grid and Random Search

In this assignment, we have also used random search and grid search to optimize the hyperparameters, and then we have compared the results with SMBO optimizing SVM and Adaboost.
For all the three considered datasets, SMBO optimizing SVM shows similar accuracies to the ones obtained with both random and grid search, whereas both random and grid search outperform SMBO when optimizing Adaboost. It's a smaller difference for datasets $mfeat-karhunen$ and *semeion* (only of 0.01), but it is more significant on the *har* dataset, where random and grid search reach an accuracy of about 0.99, whereas the Adboost optimization shows an accuracy of only 0.85. Considering the training time, we can notice that not only SMBO optimizing Adaboost is affected by the dimension of the datasets, but also grid search, which, since it evaluates all combinations of hyperparameters in a predefined grid, result in being computationally inefficient for large datasets.