Advanced ML - EX2 - Final Report

In the following report we will summarize our efforts, with regards to the first exercise in the Advanced ML course, as reflected in the attached notebook. The exercise mostly focused on GBRT and Adaboost. We have implemented both of them and did some experiments around their hyper parameters, on data we have generated ourselves.

First,we have generated two synthetic datasets, both consist of 1,000 samples. The first one has 10 features, 5 of which have significant impact on the target variable. The other 5 are linear combinations of these features. The second dataset has 5 features, of which 3 are informative and 2 are linear combinations of the informative features. We splitted both datasets to train (80% of the data) and test (20% of the data).

Then, we moved on to generate our `evaluate_model` function, which will help us throughout the rest of the exercise. This function is designed to evaluate the performance of the two different models by training them on the training data and computing various performance metrics on the training and test sets. The performance metrics we've used are accuracy, precision, recall and F1 score.

Next, we move on to implement GBRT - Gradient Boosted Regression Trees. We have chosen to implement the binary cross entropy loss, between the ground truth and the predicted probability. Inside the `fit` method, a weak regression tree is trained on the residuals. The tree is trained with the squared error criterion, where the maximum depth is a parameter. The predicted probabilities are updated by adding the product of the learning rate (another parameter) and the prediction of the input data. Each tree is stored in a list, for future use. Our `predict_proba` method iterates through the trees and aggregates their predictions by adding the product of the learning rate and each tree's prediction. The results are then passed through a sigmoid function and we get the final prediction.

Initially, we have used default parameters of n_estimators = 100, learning_rate = 0.1, tree_depth = 3. The initial results showed somewhat better performance than dataset 2, with higher accuracy compared to dataset 1, but lower F1 score.We have tested several hyper-parameters, to see their effect on the test set (which was used as a validation set).

On both datasets, we experimented with the number of trees we are using, the learning rate and the tree depth. On the first dataset, we have achieved optimal results with appx. 800 trees, maximal tree depth of 3 and learning rate of 0.4. The optimal accuracy on the test set is 0.9, the precision was 0.933, recall 0.883 and F1 score 0.907, which is a relatively big jump from the initial results we've documented. On the second dataset, the optimal number of trees was 500, the optimal tree depth was 6 and the optimal learning rate was 0.1. The best test accuracy we got was 0.91, the best test precision was 0.93, the best test recall was 0.89 and best F1 was 0.91. Generally speaking, perhaps since the second dataset is simpler in terms of informative features, the performance on it was better, compared to dataset 1.

Later, we have implemented AdaBoost, fitted for a binary class of 0 and 1.. We divided the class into several key functions that run within the `fit` function. Amongst them, a method for initiating and training each of the stumps individually, calculates the weighted error and the step size, and then re-updates the weights. The weights are updated using the current `amount_of_say` and then normalized by the sum of the weights.The `predict_proba` method is straightforward, and simply iterates over all estimators, and sums the product of the stump with its relative weight.

The only hyper-parameter we can directly change in AdaBoost is the number of weak learners, and so this was the only experiment we have conducted. Our benchmark was set by the same number of regressors we tested in the GBRT part - 100 weak learners. On dataset 1, the benchmark for precision and F1 on the test set was 0.88, and for dataset 2 the benchmark for test accuracy is 0.91 and for test F1 is 0.91.

Upon testing it on dataset 1, we saw that after 300 stumps the model starts to overfit - the training metrics continue to improve (and nearly reach 1 at 1,000 stumps) and the test metrics start to decline. After training with the optimal number of weak learners, the test accuracy and F1 scores changed to 0.89, the test precision was 0.9 and test recall was 0.87.

The findings were consistent on dataset 2, only that after 400 stumps the test performance started to decline. This differs from what we've seen in GBRT, where dataset 1 required more trees than dataset 2. In the final results, the test accuracy and F1 score were 0.93, the test precision was 0.95 and the test recall was 0.91. This, again, is president with what we've seen in the GBRT part - probably because the second dataset is less complex, the performance on it is better compared to dataset 1.

To sum up, in this exercise we have implemented and tested GBRT and AdaBoost. We created datasets to test them on, implemented both algorithms, while relying on the sklearn tree algorithms, and experimented with their most prominent hyper parameter. In all cases, we were able to improve the algorithm's performance, after adjusting the hyper-parameters.