Exercise 6

Deadline: 04.07.2023, 16:00

This exercise is devoted to tree-based machine learning methods. Use the stubs provided in file `tree-methods-stubs.ipynb` (see `https://tinyurl.com/HD-EML-tree-stubs-npynb` or the external link on MaMPF) as a starting point for this homework.

Regulations

Please hand in your solution as a Jupyter notebook `tree-methods.ipynb`, accompanied with `tree-methods.html`. **It is important that you stick to these file naming conventions!** Zip all files into a single archive `ex06.zip` and upload this file to MaMPF before the given deadline.
Moreover, please set your **Anzeigename/display name** and **Name in Uebungsgruppen/name in tutorials** in MaMPF to your real name, which should be identical to your name in `muesli` and make sure you **join the submission** of your team via the invitation code before the submission deadline. Check out `https://mampf.blog/handing-in-homework-assignments` for instructions.

1 Classification and Regression Tree (16 points)

Complete the implementation of the classes `ClassificationTree`, `RegressionTree` and its base class `Tree` from the stub notebook. The relevant locations are marked with `... # your code here`. Evaluate the performance of these models by 5-fold cross-validation on the 3s and 9s of sklearn's `digits` dataset and comment on your results. Recall the following rules for tree construction:

1. The permitted class labels are passed to the constructor of `ClassificationTree`. In contrast, the regression tree always expects ground-truth responses $+1$ and -1 (cf. exercise 05).
2. All splits are axis-aligned, i.e. we split the data according to the value of a single feature in every split node. Split nodes must store the index of this feature and the split threshold.
3. To decide about the best split, only $D_{\text{try}} \leq D$ candidate features are eligible in every node, with $D_{\text{try}} = \sqrt{D}$ being the default. In each call, the function `select_active_indices()` returns another random subset of size D_{try} of the indices $0...(D-1)$.
4. For each feature considered, the candidate splitting thresholds are located in the middle between adjacent feature values after sorting. The function `find_thresholds()` returns the list of these thresholds.
5. The function `compute_loss_for_split()` returns the loss of a proposed split or `float('inf')` if the split is not permitted, i.e. a child would contain fewer than `n_min` instances. `n_min` is a hyperparameter with default value 10 – you may play with this value. Use the squared loss for regression and the entropy or Gini impurity for classification (see the lecture for detailed explanations).
6. When the best split is accepted, the function `make_children()` creates the split's child nodes and assigns the appropriate instances (i.e. their features and responses) to them.
7. When a node cannot be split any more (either because the loss doesn't improve, or no split is permitted), the function `make_leaf_node()` turns the present node into a leaf by computing and setting its `prediction` attribute (see the lecture for more details).
8. For cross-validation, see exercise 01.

If implemented correctly, training of a tree should take less than 10 seconds.

2 Classification and Regression Forest (8 points)

Repeat the same tasks with the classes `RegressionForest` and `ClassificationForest`. Recall the following rules:

1. Each tree in the forest is trained with a different bootstrap sample created by the function `bootstrap_sampling()`. A bootstrap sample is obtained from the original training set (of size N) by randomly selecting N instances *with replacement* (see the lecture for more details). On average, about 36.7% of the original training instances will be missing from any given bootstrap sample. This ensures diversity of the trees in the forest.
2. The prediction of the forest is calculated from the predictions of the individual trees by a suitable voting or averaging scheme.

3 Multi-class Classification Forest (8 points)

Extend `ClassificationTree` and `ClassificationForest` so that they can handle an arbitrary number of classes (if not already implemented so). Use 5-fold cross-validation on the entire `digits` dataset (i.e. all ten digits), plot the confusion matrix and comment on your results.

4 Multi-class Regression Forest (8 points)

Use `RegressionForest` to implement ten one-against-the-rest classifiers for the entire `digits` dataset. In each tree, make sure that the bootstrap dataset is balanced between the current target class and the "rest". Extend `bootstrap_sampling()` accordingly. The ensemble returns the class whose one-against-the-rest classifier has the highest score, or `'unknown'` if all scores are negative (i.e. all classifiers vote for the "rest"). Use 5-fold cross-validation, plot the confusion matrix and comment on your results.