Exercise 4

Deadline: 28.11.2018, 15:00

This exercise will introduce you to tree-based machine learning methods. Use the stubs provided in file `tree-methods.ipynb` on Moodle as a starting point for this homework.

Regulations

Please hand-in your solution as a jupyter notebook `tree-methods.ipynb`, accompanied with exported HTM `tree-methods.html`. Zip all files into a single archive with naming convention (sorted alphabetically by last names)

`lastname1-firstname1_lastname2-firstname2_exercise04.zip`

or (if you work in a team of three)

`lastname1-firstname1_lastname2-firstname2_lastname3-firstname3_exercise04.zip`

and upload it to Moodle before the given deadline. We will give zero points if your zip-file does not conform to the naming convention.

1 Density Tree and Decision Tree (20 points)

Complete the code of the classes `DensityTree` and `DecisionTree` in the stub, following the explanations given in the lecture and in code comments. The optimal splits of a *density tree* shall minimize the leave-one-out error of the resulting children. For a given node l, the leave-one-out error is defined as

$$\text{looErr}_l = \frac{N_l}{NV_l}\left(\frac{N_l}{N} - 2\frac{N_l - 1}{N-1}\right)$$

where N_l and V_l are the number of instances in node l and its volume, and N is the total number of instances for the class under consideration. The region spanned by node l is represented by the vectors m_l and M_l of lower and upper bounds respectively, so that $V_l = \prod_{j=1}^{D}(M_{lj} - m_{lj})$.

In case of a *decision tree*, the optimal splits shall minimize the Gini impurity of the resulting children. The Gini impurity of node l is defined as

$$\text{Gini}_l = N_l\left(1 - \sum_{k=1}^{C}\frac{N_{lk}^2}{N_l^2}\right)$$

where N_l is the total number of instances in node l, and N_{lk} are the number of instances of class k in l.

To save computation time, only a random subset of size $D_{\text{try}} = \sqrt{D}$ of the features shall be considered when searching for the optimal split in each node. Note that different subsets must be selected in every node. Function `numpy.random.permutation()` and "advanced indexing" (see `https://docs.scipy.org/doc/numpy/reference/arrays.indexing.html#advanced-indexing`) will be helpful here.

Candidate thresholds shall be placed in the middle between consecutive feature values

$$t_{i'j} = \frac{X_{[i]j} + X_{[i+1]j}}{2}$$

provided that $X_{[i]j}$ and $X_{[i+1]j}$ are different. The brackets in $X_{[i]j}$ denote sorted order w.r.t. to feature j. If the two feature values are the same, no threshold shall be placed there. Describe in a comment why this is necessary.

2 Evaluation of Density Tree and Decision Tree (6 points)

Once again, use the digits dataset provided by sklearn. This time, we will not split the data into training and test sets and just measure the training error (otherwise, the dataset would become too small for density estimation). Train a generative classifier using 10 instances of `DensityTree` and a discriminative classifier using one instance of `DecisionTree`. For each method, plot the full training error confusion matrix and comment on the results.

3 Density Forest and Decision Forest (8 points)

Complete the code of the classes `DensityForest` and `DecisionForest` in the stub, following the explanations given in the lecture and in code comments. To make the trees in the forest independent of each other, create a new bootstrap training set for each tree. That is, create new training sets with the same size as the original one by random instance selection *with replacement*. Functions `numpy.random.choice()` and "advanced indexing" are useful here.

The ensemble prediction shall be the average of the individual tree predictions.

4 Evaluation of Density Forest and Decision Forest (6 points)

Train a generative classifier using 10 instances of `DensityForest` and a discriminative classifier using one instance of `DecisionForest`, with each forest consisting of 20 trees. For each method, plot the full training error confusion matrix and comment on the results.