Introdution to the files

PUID: 0030193826

November 11, 2018

I follow the exact algorithm provided by the paper, and the code should be attached. We take the linear pattern case as a example. Training the ID3 tree is executed by the file *train.py*. There are 4 classes and 19 functions in this file. A biref look at them is listed as follows:

- The class `Event` and function `read_data`() are used to load the events (or observations) from the training data file *TrainLinearData.root*. There are 2048 events used for training.
- The function `bin_combinations`() provides the sliding window subsets within the given window size.
- The functions `find_centroid`(), `point_point_dist`(), and `find_split_point`() are to get the split points at the beginning. Then, the functions `cal_split_plane`() and `find_best_plane`() calculate the first best split hyperplane from those split points.
- The following two functions `refine_split_points`() and `refine_best_plane`() refine the best split plane.
- The two functions of the rest, `find_best_split_node`(), `build_ID3_tree`(), aim to build the decision tree greedily by the ID3 algorithm.

Eventually, the output file is saved as *ID3_tree.root* by the `main`() function.

The file *prune.py* does pruning. Also, the following can provide a glance on the functions in the file.

- `read_data`() loads the dataset for pruning, and `read_ID3_tree`() loads the trained ID3 decision tree.
- `cal_frac_accu`() calculates the number of events (n_v) having the same label as the node for each node (v), recursively.
- `find_best_offsp_weighted_accu`() can calculate the most correct decisions ($n_{T(v)}$) that can be made by (any part of) the sub-tree ($T(v)$) rooted at each node (v), recursively.
- `prune_sub_tree`() and `find_prune_nodes`() do prune the sub-trees after their root node v, iff $n_{T(v)} \leq n_v$.

The post-pruned decision tree is saved as the file *postpruned_ID3_tree.root*.