Homework assignment # 3

Due:

IMPORTANT NOTE #1 When you are asked to explain something *concisely*, *clearly*, and/or *briefly*, *please do exactly that.* Lengthy answers not to the point will not earn you extra credits but will cost you more time. Also, *please write neatly and legibly.* Poor penmanship that is hard to decipher will not earn you any sympathy points.

IMPORTANT NOTE #2 You can dicuss the problems with your fellow students, TAs and instructor, but all answers and codes written down and turned in must be your own work.

These notes apply to all assignments in this class.

The task here is to find the computational complexity of training a tree classifier using the twoing principle (binary split, with patterns in the same class kept together in a split) and candidate splits are based on a single feature. Suppose there are n sample patterns and c classes, and each class has n/c patterns that are d-dimensional. Proceed as follows:

1. How many possible non-trivial divisions into two super-categories are there at the root node? Note that the split does not have to be balanced (i.e., the left and right branches can have different numbers of classes/samples).

2. For any one of these candidate super-category divisions, what is the computational complexity of finding the best split that minimizes the entropy impurity?

3. Use your results from parts above to find the computational complexity of finding the split (with the smallest entropy impurity) at the root node.

   From here onward, we will assume that all splits divide the patterns into subsets of about equal sizes. That is, each split will result in about the same number of classes (differ by at most 1) in the left and right branches.

4. Suppose that the node division will continue until each leaf node corresponds to patterns of a single class. In terms of the variables given, what will be the expected number of levels of the tree?

5. Naturally, the number of classes represented at any particular node will depend upon the level in the tree; at the root all c categories must be considered, while at the level just above the leaves, only 2 categories must be considered. (The pairs of particular classes represented will depend upon the particular node.) State some natural simplifying assumptions, and determine the number of candidate classes at any node as a function of level (root is level 0, children of root is level 1, grandchildren of root is level 2, etc.) You may need to use the floor or ceiling notation, $\lfloor x \rfloor$ and $\lceil x \rceil$, in your answer.

6. Use your results from above and the number of patterns to find the computational complexity at an arbitrary level L.

7. Use all your results to find the computational complexity of training the full tree classifier.

8. Suppose there $n = 2^{10}$ patterns, each of which is $d = 6$ dimensional, evenly divided among $c = 16$ categories. Suppose that on a uniprocessor a fundamental computation requires roughly 10^{-10} seconds. Roughly how long will it take to train your classifier using the twoing criterion? How long will it take to classify a single test pattern?

Programming exercise Use the decision tree algorithm discussed above on the Iris data set https://archive.ics.uci.edu/ml/datasets/Iris. The data set contains three classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are not linearly separable from each other.

Your decision tree (DecisionTree) should take the following parameters:

- *x*: a numpy array of size $n \times k$, with n being the number of training samples, and k being the number of features per training sample
- *y*: a numpy array of size $n \times 1$ of the ground-truth class IDs, with n being the number of training samples
- *depth*: maximum tree depth (an integer). The root level will be 0, the children of the root will be at level 1, and the grandchildren of the root will be at level 2, etc. Depth of d means that you can have nodes all the way to level d.
- *tree*: the decision tree (in any data structure/format that you design, as long as your program can read it back and reuse it).

x, y and *depth* will be supplied in the training phase and x and *tree* will be supplied in the validation (testing) phase. All other specifications are invalid. All parameters not specified in the assignment (e.g., impurity measure) are for your own design. Again, you can add more parameters to the end of the above list if you like. You should return *tree* in the training phase and y in the testing phase. Note that you should not apply twoing principles in this assignment as it does not work well (trees will often be very shallow) when the number of classes is small.

Furthermore, for this assignment, you should experiment with n-fold validation, with $n = 5$ by separating the data sets into five parts (10 instances each for each class) and use four parts for training and the remaining part for validation. You will do the training and validation 5 times (with different parts held back for testing) and then average the training and validation accuracy as the final results. Experiment with decision trees of different depths and provide data (in jpeg or png format) on training and validation errors as functions of the

maximum allowed tree depth, in addition to your codes. What do you observe from the results? Explain why decision trees are seldom used alone, but are used as a collection (i.e., a random forest) through ensemble learning.