CSE 455/555 Spring 2013 Homework 1: Tree

Jason J. Corso
Computer Science and Engineering
SUNY at Buffalo
jcorso@buffalo.edu

This assignment does not need to be submitted and will not be graded, but students are advised to work through the problems to ensure they understand the material.

You are both allowed and encouraged to work in groups on this and other homework assignments in this class. These are challenging topics, and working together will both make them easier to decipher and help you ensure that you truly understand them.

Finally, I am sure fully functioning decision tree and forest code can be found on the internet and referenced or copied into your own code, but I would suggest that you work out as much of this assignment as possible on your own—simply referring to someone else's code will not help you learn how these algorithms function the same way writing the code yourself will.

Programming Problem: Decision Trees and Random Forests

This is the first programming question of the term and it requires you to work with decision trees and random forests in Python.

For starters, you need to download and get familiar with the `prpy` package that has been developed for this course. The `prpy` code, as well as some example usage, can be found on the site at `http://www.cse.buffalo.edu/~jcorso/t/555code/`.

Next, download the homework starter source file `http://www.cse.buffalo.edu/~jcorso/t/CSE555/files/rfdigits_hw1.py`.

Finally, download the dataset from the website or locate it on the CSE network:
`http://www.cse.buffalo.edu/~jcorso/t/555code/data/555_mnist_data.tar.bz`. The datafiles are also available on the CSE network at `/projects/jcorso/CSE555/555_mnist_data` that you can use without actually copying them locally into your account. This is a subset of the LeCun's MNIST handwritten digit recognition dataset containing just the digits 0 through 5. The full dataset is available at `http://yann.lecun.com/exdb/mnist/`. The dataset is split into training and testing pictures. Do all training on the training pictures, and reserve the testing pictures for classification.

I will not provide a tutorial on the prpy package or the assignment source files, other than what has been described in class. You can read the documentation in the source files and the examples at `http://www.cse.buffalo.edu/~jcorso/t/555code/code/examples/` to get a better idea of what is going on.

1. The first part of the assignment has you becoming acquainted with the existing Decision Tree code and tools. Use the prpy code and provided examples to generate a 4 class 2D dataset (it will likely be helpful to plot this dataset so you can visualize the resulting pattern recognition problem). Learn a decision tree (using the methods in trees.py) and compute the accuracy of the decision tree on your data set (see `pr.datatools.accuracy`). Plot the decision regions using the `pr.trees.testGrid` function. Observe the results, data and decision regions, and see if you can describe why the method performed as it did.

2. Next, implement a method like "impurity_entropy" that calculates Gini impurity instead and retrain the decision tree using this impurity measure in place of entropy. Compute the accuracy. What has changed?

3. Next, you are to implement a Random Forest classifier for classifying grayscale images of handwritten digits. For simplicity, I have provided a `rfdigits_hw1.py` starter file that contains the full skeleton of the random forest code (both learning and classification) as well as the functionality to load in the data and compute features on it.

   You should implement the missing parts of the code as specified in the file. Two of the parts are for the classification side and one is for the learning side. All three are important to understanding the algorithm.

   Just like in the Amit and Geman digits example given in class (possibly next class, depending on when you're reading this), the features that have been defined in this code are too many to enumerate. So, during the randomized learning of the tree, you will have to dynamically instantiate a random set of, say 100, features and learn the best query based on them. Then, at the next node, you instantiate a new random set of features and so on. The actual features that we will use are simple weighted sums of pixel values (all of the images are 28×28 grayscale). The class `KSum_Feature` has been provided for you to easily compute these features. Finally, the class `DTQuery_KSum` has been provided to encapsulate a query based on these features; you do not need to use this one if you don't want, but it is recommended.

   You should complete the `rfdigits_hw1.py` file, and execute it (by typing, for example, `python rfdigits_hw1.py` assuming your `PYTHONPATH` is set properly. Running this script will load a subset of the data, train the tree, and then compute the accuracy on the training set as well as the accuracy on the provided testing set.

   Consider what happens when you run the `rfdigits_hw1.py` file; i.e., what are you accuracies? How do the accuracies vary as you change the parameters of the tree (what if you use a `featureDepth` of 10 or 100, how does this change the accuracy?).

   You are encouraged to explore the classifier and the problem further to, for example, display images of digits that were misclassified or visualize the paths down a tree given the selected type of feature.