CSE 417T: Homework 4

Due: April 13 (Monday), 2020

Notes:

- Please submit your homework via Gradescope and check the submission instructions.
- You may work in groups of up to two persons on this homework. Each group only needs to submit one copy of the homework. Check the following link on how to submit group assignments on Gradescope. Make sure you follow the instructions and assign team members correctly.
  `https://www.youtube.com/watch?v=a6DERS94qPY&feature=youtu.be&t=32s`
- Please download the following files for this homework.
  `http://chienjuho.com/courses/cse417t/hw4/hw4_files.html`
- Homework is due **by 11:59 PM on the due date.** Remember that you may not use more than 3 late days on this homework, and you only have a budget of 7 in total.
- Please keep in mind the collaboration policy as specified in the course syllabus. If you discuss questions with others you **must** write their names on your submission, and if you use any outside resources you **must** reference them. **Do not look at each others' writeups, including code.**
- Please comment your code properly.
- There are 2 problems on 2 pages in this homework.

Problems:

1. (50 points) The purpose of this problem is to write code for bagging decision trees and computing the out-of-bag error. You may use matlab's inbuilt `fitctree` function, which learns decision trees using the CART algorithm (read the documentation carefully), but do not use the inbuilt functions for producing bagged ensembles. In order to do this, you should complete the stub `BaggedTrees` function. Note that it only returns the out-of-bag error. You may want to use other functions that actually construct and maintain the ensemble. You may assume that all the **x** vectors in the input are vectors of real numbers, and there are no categorical variables/features. You will compare the performance of the bagging method with plain decision trees on the handwritten digit recognition problem (the dataset is in `zip.train` and `zip.test`, available from `http://amlbook.com/support.html`. [1])

   We will focus on two specific problems – distinguishing between the digit one and the digit three, and distinguishing between the digit three and the digit five.

[1] Check the links to "training set" and "test set".

a Complete the implementation of `BaggedTrees`. You may choose any reasonable representation that you wish; the two strict requirements are that you plot the out-of-bag error as a function of the number of bags from 1 to the number specified as input (`numBags`), and that you return the out-of-bag error for the whole ensemble of `numBags` trees. Include the plots (with clearly labeled axes) in your writeup, and, of course, submit your code.

b Run the provided `OneThreeFive` script, which creates training datasets based on the one-vs-three and three-vs-five cases we are interested in, and calls both the in-built decision tree routine and your bagging code, printing out the cross-validation error for decision trees and the OOB error for your bagging implementation. Report the results in your writeup.

c Now, learn a single decision tree model for each of the two specified problems (one-vs-three and three-vs-five) on the training data, and test their performance on `zip.test` what is the test error? Similarly, learn a single ensemble of 200 trees on the training data for each of the two specified problems and test the performance of the ensembles on the test data. Report your results.

d Summarize and interpret your results in one or two concise paragraphs as part of your writeup.

2. (50 points) Implement AdaBoost using decision stumps learned using information gain as the weak learners (you may use the `fitctree` function to implement the weak learner. Look at the "deviance" split criterion), and apply this to one-vs-three and three-vs-five problems (as described in Question 1) on the `zip.train` and `zip.test` data. In order to do this, you should complete the stub `AdaBoost` function. Graphically report the training set error and the test set error as a function of the number of weak hypotheses, and summarize and interpret your results.

For code submissions, please submit all files that will be needed to return the results. You are free to modify `OneThreeFive.m` and/or include additional files. The only requirements are (1) the results can be obtained by running `OneThreeFive.m` and (2) the two stub files are filled in. It is okay to have additional files. It is also okay if you want to modify the input/output specification slightly, but please make sure you commented it appropriately.