CSE 417A: Homework 6

Due: November 17, 2015

Notes:

• You may work in pairs on this homework. Each of you only needs to turn in one copy in total if you work in a pair. Please do not collaborate with anyone other than your partner on this homework.

• Instructions for how to get files from the SVN repository are available on the course website and on Piazza.

• There is no autograder for this homework.

• Homework (in hardcopy) is due at the beginning of lecture. In addition, your code submissions must also be timestamped before lecture begins.

• Please comment your code properly.

• There are 3 problems on 2 pages in this homework.

Problems:

1. (45 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 available in your SVN repository. 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 from HW3 (the dataset is in zip.train and zip.test – the same files as in HW3, available from http://amlbook.com/support.html). 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. Here are the steps for this problem:

(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, commit 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. (10 points) Now use Matlab’s inbuilt TreeBagger functions (and any extensions / options thereof) to learn random forests for these two classification problems. Report the OOB accuracies and the test accuracies you get with forests of 200 trees, and compare with those for bagged trees. Are the results in line with what you would expect?

Note: You do not need to submit any code for this part, just mention how you implemented it (e.g. what parameters you used) and discuss the results in your writeup. In order to get the results, you should read the matlab documentation for TreeBagger carefully.

3. (45 points) Implement AdaBoost using decision stumps learned using information gain as the weak learners, and apply this to the one-vs-three and three-vs-five problems described in Question 1 on the zip.train and zip.test data. In order to do this, you should complete the stub AdaBoost function available in your SVN repository. You may add and use other files as needed, but be sure to check them in to your repository (you do need to submit your code for this problem). 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.