Problem Set 6
Due on Friday, November 18, 2016 at 11:55 pm

How to Submit

Create one zip file (.zip) of your code and plots and submit it via `ilearn.ucr.edu`. This zip file should have `parta.m`, `partb.m`, `partc.m`, `partd.txt`, plus any other code that you have written.

Do not supply any directories in your zip file. Each file should (in comments) list

- Your name
- Your UCR student ID number
- The date
- The course (CS 229)
- The assignment number (PS 6)

Ensemble Classifiers

Supplied Code: To help with this assignment, code is supplied to learn, evaluate, and print a decision tree. The relevant files are `traindt.m`, `traindtw.m`, `dt.m`, and `printdt.m`. They have help comments, so typing `help dt` at the command prompt will explain how to use the command. `traindt.m` builds a decision tree with no pruning. It defaults to using the Gini index for its splits. You should keep this default. To build a tree to a maximum depth of 3, you should call

```
t = traindt(X,Y,3)
```

If you have weights on the points (in a vector `alpha`), you should call

```
t = traindtw(X,Y,alpha,3)
```

The tree returned (`t` in this case) is a cell array. It cannot be stored in a vector and needs to be stored on its own or in a cell array (if you want to group it with other things).

For the bagging and boosting classifiers, you will need to run on the same data repeatedly, with increasing number of classifiers (rounds). When doing this, do not restart your learning from scratch for each new value of the number of elements in the ensemble. To save time (and make the plots have less variation), use the old (smaller) result as the starting point for the next compuations. For example, if you already have the result for boosting 120 trees, to get the result for boosting 140 trees, just add on another 20 trees (starting with the correct alpha weights per data point, of course).

part a: Bagging [2 points]
On the `class2d` dataset, plot the decision surface (using `plotclassifier`) for bagging decision trees. In a single plot (using four `subplot`s, each labeled) plot the result using 1, 10, 100, and 1000 decision trees. Do this three times (therefore, three plots, each with four subplots), changing the maximum tree depth to be 1, 2, and 3. Call the code that performs this `parta.m`.

part b: Boosting [3 points]
Now do the same as part a, but for boosting (the boosting algorithm in class: Adaboost). Call the code that performs this `partb.m`.

part c: Spam [3 points]
The supplied files `spamtrain.ascii` and `spamtest.ascii` have training and testing data for a spam

classification task (see `http://archive.ics.uci.edu/ml/datasets/Spambase` for more details). The *last* column (58) is the label. The others are the features.

Make two plots: one for bagging and one for boosting. Each plot should plot the training and testing error as a function of the number of rounds (trees added). Select the number of rounds as given by the Matlab expression `floor(logspace(0,3,10))` and plot using `semilogx`.

Run bagging and boosting each three times (producing six curves: three training errors and three testing errors), changing the maximum tree depth to be 1, 2, and 3. Make the training errors dashed lines and the testing errors solid lines. Set the `linewidth` to be equal to the tree depth. Label the axes and title the plots to make it clear which is for bagging and which is for boosting.

Call the code that performs this `partc.m`

part d: So What? [3 points]
Explain the differences you see (using what you can tell from all three parts, above) between boosting and bagging. How do the methods relate to the bias and variance of the underlying classifier? How do training and testing errors compare?

Place your answer in a file called `partd.txt`