Problem Set 7
Due on Friday, December 5, 2014 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 a single script named `ps7.m` that runs all of the examples. It should also contain the code necessary to run these samples. Finally, it should contain a single pdf file (ps7.pdf) that has your plots.

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 7)

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 requires a function to build the tests. `entropysplit.m` supplies such a function. To build a tree to a maximum depth of 3 with a minimum number of examples per leaf of 5, you should call

```
t = traindt(X,Y,3,@splitentropy,5)
```

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

```
t = traindtw(X,Y,alpha,3,@splitentropy,5)
```

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).

Code is also supplied to fit a logistic regression model with L1 regularization. The file is `trainlogregL1.m`. It also has a help section. Note that it returns a vector 1 longer than the input xs. This is because it has the bias at the *end* of the returned vector.

For the bagging and boosting classifiers, you will need to run on the same data repeatedly, with increasing number of classifiers. When doing this, you do not need to 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. Of course, you will have to be careful about getting the weights right when you restart.

part a: Bagging
On the `class2d` dataset, plot the decision surface for bagging decision trees. In particular, plot (in a single plot, using four subplots, each labeled) for using 10, 20, 50, and 100 decision trees. Your decision trees should be trained with a maximum depth of 3 and a minimum number of data points per leaf of 5.

part b: Refit Bagging
Take the last bagging result (of 100 trees) from part a and refit the coefficients on each bag using logistic regression with an L1 penalty. Again, make four labelled subplots of a single figure for L1 penalty coefficients of $\lambda = 0, 2, 5, 10$. On each plot, note the number of trees used (non-zero weights).

part c: Boosting
Now do the same as part a, but for boosting (the boosting algorithm in class: Adaboost). Use the same number of trees and same parameters for the base decision tree classifier.

part d: Spam
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 a single plot of all three methods above in which you plot the *testing error rate* as a function of the number of trees. For boosting and bagging, vary the number of trees from 20 to 200 in increments of 20. For refitting of the bagging result, use the final 200-tree bagging result and refit it with lambda taking on values from 0 to 5 in increments of 0.5.

For this data set, let the maximum depth of the base decision tree be 6 and the minimum number of elements in a leaf to be 100.

Warning: not all methods perform equally well.

part e: Spam
Explain the plot in part d.