CS4495 Fall 2011 — Computer Vision
Project 6: Classificationw

Officially Due: **Thursday** Dec 8 - 4:30pm
but
Can be handed in until: **Monday** Dec 12 - 11:55pm

In this project you will code up your own `AdaBoost` algorithm for doing classification. Because this is the last project of the semester and there is limited time left, we will not classify real images but instead work on a "toy problem" of some data given in two dimensions.

As always, a reminder as to what you hand in: A Zip file that has

1. Output: Images (either as JPG or PNG or some other easy to recognize format) clearly labeled using he convention Proj<number>-<question number>-<question sub>-counter.
2. Code you used for each question. It should be clear how to run your code to generate the results. Code should be in different folders for each main part with names like Proj2-1-code. For some parts — especially if using MATLAB — the entire code might be just a few lines.
3. Sometimes the question asks for output other than an image, such as a matrix or a written response — put all non-image output in a readme.txt file in the code folder.
4. Zip file should be name <gt_username>.zip, where <gt_username> is replaced by your Georgia Tech username as it appears in T-Square.

This project uses files stored in the directory `http://www.cc.gatech.edu/~afb/classes/CS4495-Fall2011/projectfiles/project6/data` There are a two sets of data, a training set and a testing set. Each set has data points for class A and class B. These are called (ca you guess?): `trainA.txt`, `trainB.txt`, `testA.txt`, and `testB.txt`. These are text files (duh?) where each row of two numbers is a new data point in two dimensions. (We told you this was a toy problem.)

1 Plot 'em

To make sure you've got a handle on the data (and some plotting you will need to do later), create a scatter plot of the two classes. use distinct markers and colors for the two classes. Make the x axis be the first dimension variable and the y axis the second.

1.1 Write code to read in the data points from both the training and testing set. For each set create a scatter plot for the two classes (uh, you'll have two scatter plots).

Output: The code, and the 2 scatter plots of the training and testing data.

2 Finding stumps

Recall that the boosting methods we discussed in class creates what is called an *ensemble* of weak learners which when combined with appropriate weights gives a good classifier. The weak learners are some not-perfect classifiers that give some information.

A very simple form of a weak learner is the so-called *decision stump* which is a *decision tree* of only one node. (Get it? A stump is a very short tree!?!?) In our case the stumps will be a simple test of either the first or second dimension of the data points (call these dimensions x_1 and x_2. So for example, a stump might be "`if` $x_1 < .4$ `decide class 'A' otherwise decide 'B'`".

Because our data is very simple we can exhaustively look for the minimum weighted-error stump by starting at the smallest x_1 (or x_2) value and incrementing by small steps. At each point we can evaluate how many points would be correctly classified if we were to use only that stump. We say "weighted error" because for boosting we will be increasing the weights of incorrectly labeled data. Given a decision stump, the weighted error is the sum of weights of the incorrectly classified points.

2.1 Write code to search for the best stump in a given dimension for a weighted set of data points. Apply this to the training data set, for now assuming a uniform weight. Find both the best x_1 and x_2 stumps. Then, plot these decision boundaries on scatter plot of the training data. The x_1 stump can be drawn as a vertical line at the decision point; the best x_2 is a horizontal line.

Output: The code and the scatter plot with both stump lines drawn.

3 AdaBoost

You can now implement the AdaBoost algorithm. There are several places where you can get the details. There is a nice algorithm box in the Szeliski book (page 665 in the PDF version). The slides at `http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf` are very nice and the algorithm is fully illustrated on page 11 (and others). The math there is a little different than in Szeliski but they are both legal boosting methods. The bottom line is that the incorrectly classified examples get more weight relative to the correctly classified ones.

3.1 Write the code for the AdaBoost algorithm. For finding the best weak learner h_t for each iteration t, find the best stump (could be in either dimension x_1 or x_2). For a given number of rounds T, you will get a final classifier that is a weighted sum of T weak learners. You should be able to create a table of how many errors your final classifier makes as you vary T. You will train your classifiers on the training data `trainA` and `trainB`.

Output: Code for AdaBoost and a table that shows how many errors on the training data for T equal to 4, 8, 16, and 24. Also, for T equal to 4, you will have 4 stumps. Plots them on a scatter plot of the training data. (If some of the stumps are identical (it can happen) plot the first 4 distinct stumps.)

3.2 Now test each of your final classifiers($T = \{4, 6, 16, 24\}$) on the test set.

Output: The table that shows how many errors on the test data for T equal to 4, 8, 16, and 24.