COMS 4771: Machine Learning
Homework 3, due April 5.

Note: In your write-up please only provide the specific information requested. In particular, no code need be submitted.

This homework will use the same data sets that you used in the last homework, from the file `spam.mat` which can be found on Canvas or the course webpage. Loading this file into `MATLAB` will generate two matrices: `train_spam`, the training set, with 3601 labeled examples, and `test_spam`, the test set, with 1000 labeled examples. Each example represents an email with 57 features and a binary label indicating whether the email is spam (label = 1) or not (label = 0). A description of the data can be found at `https://archive.ics.uci.edu/ml/datasets/Spambase` with detailed descriptions of the features at `https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names`. The rows of the data set found there have been randomly permuted to generate the train and test sets.

In this homework, you will experiment with AdaBoost using decision stumps as the base learner. Decision stumps are extremely simple classifiers which use a single feature and a threshold to make their prediction: in other words, they're depth 1 binary decision trees. To train a decision stump, we compute, for each feature i, and for each threshold value θ among the values of feature i, the cost of the decision rule which splits the training set using into two parts (examples with $x_i > \theta$, and examples with $x_i \leq \theta$) and using the majority label as a prediction in each part. The cost is simply the number of mistakes made by this decision rule (this corresponds to classification error as the measure of uncertainty: don't worry about Gini index and entropy in this homework). To use decision stumps in AdaBoost, we need to generalize this training method to allow weights on examples.

Part 1 of assignment: (weightage: 1%.) Give pseudocode for training a decision stump when traininig examples are given weights. Use the same tie-breaking rules as the last homework: in case there are two features x_i and x_j that both yield minimum cost decision stumps, break the tie by choosing the lower indexed one (i.e. i if $i < j$). Also, if for any leaf, the costs of predicting 1 and 0 are the same, break the tie in favor of the 0 label.

Next, implement your pseudocode in `MATLAB`. Using this code, write a `MATLAB` function

```
function params = hw3_train_adaboost(train_data, num_rounds)
```

where the integer parameter `num_rounds` is the number of rounds of AdaBoost to run on the training set `train_data`, using your decision stump training code as the weak learner in each round. The output `params` is any convenient `MATLAB` data structure to represent the ensemble of decision stumps (and their weights) computed by AdaBoost.

Write another function which computes predictions for test data given the parameters computed by the `hw2_train_adaboost` function, and error rate[1] made on the test data, with the following signature:

```
function loss = hw3_test_adaboost(params, test_data)
```

Part 2 of assignment: (weightage: 3%.) Run AdaBoost using `hw3_train_adaboost` with `train_data = train_spam` and `num_rounds` $= 1, 2, \ldots, 100$. For every value of `num_rounds`, compute the training error by using `hw3_test_adaboost` on `test_data = train_spam`, as well as the test error by using `hw3_test_adaboost` on `test_data = test_spam`. Plot curves (on a single graph) of the training error and test error on the y-axis with the number of rounds on the x-axis, and include the graph in your homework submission. Also report a few specific values of the training and test errors in a table with the following format:

`num_rounds`	training error rate	test error rate
1		
2		
4		
8		
16		
32		
64		
100		

Although it is not required for this homework, you may wish to compare the numbers you get using AdaBoost with decision stumps with the numbers you got for training decision trees in the last homework. To do a fair comparison, a complete decision tree of depth d has 2^d leaf nodes, each of which can be thought of as a decision stump, so compare such a decision tree to a classifier constructed by AdaBoost running for 2^d rounds.

Hint: Since AdaBoost constructs the ensemble incrementally, the output `param` computed by running AdaBoost for 100 rounds can also be used to reconstruct the output of AdaBoost for any fewer number of rounds.

Part 3 of assignment: (weightage: 2%.) *Polynomial regression* is a technique for solving regression problems by fitting a bounded degree polynomials to the data rather than a linear function, as in ordinary least squares. The simplest case is when examples are represented a single scalar feature $x \in \mathbb{R}$. The class of regression functions, $\mathcal{F}_d$, is the set of all polynomial functions of x with degree bounded by d, where d is a parameter. Thus, the class can be defined as

$$\mathcal{F}_d = \{c_0 + c_1 x + c_2 x^2 + \ldots c_d x^d \mid c_0, c_1, \ldots, c_d \in \mathbb{R}\}.$$

Suppose we use squared loss to measure goodness of fit, i.e. for an example (x, y), if the prediction $\hat{y}$, then the loss is $(y - \hat{y})^2$. So the polynomial regression problem is to find the polynomial $f \in \mathcal{F}_d$ which minimizes training error on some training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, i.e.

$$\arg\min_{f \in \mathcal{F}_d} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

[1] Error rate is the fraction of mistakes, i.e $\frac{\#\text{mistakes}}{\#\text{examples}}$.

Give a closed form formula for solving the polynomial regression problem, ignoring any numerical issues.

Extra credit: (weightage: 2%.) We have informally argued that the AdaBoost algorithm uses the weighting mechanism to force the weak learner to focus on the problematic examples in the next iteration. In this question we will find some rigorous justification for this argument. Compute the value of

$$\sum_{(\boldsymbol{x},y)\in S} D_{t+1}(\boldsymbol{x},y)\cdot y f_t(\boldsymbol{x}).$$

Here, the notation is the same as used in the lecture slides: f_t is the classifier found by AdaBoost in round t, and D_{t+1} is the distribution computed from D_t using f_t. Based on the value you computed, what can you say about how good of a classifier f_t is when examples are drawn from D_{t+1}?

Extra credit: (weightage: 1%.) In your experiments, you may have noticed something curious about the error rates (either training or test error) when you run AdaBoost for 1 round or 2 rounds. Explain why this happens.