COMS 4771: Machine Learning
Homework 2, due February 23.

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

This homework will use the data sets 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.

Write a `MATLAB` function

```
function tree = hw2_train_DT(train_data, depth)
```

which trains a complete binary decision tree on the training set `train_data`. The parameter `depth` is a positive integer indicating the depth of the tree: for example, `depth = 1` is a tree with a root node and 2 child nodes which are leaves, `depth = 2` is a tree with a root node, 2 child nodes of the root node, and 2 child nodes each of each child node of the root node which are leaves, for a total of 4 leaves. The output `tree` is any convenient `MATLAB` data structure to represent the decision tree. I found `MATLAB cells` particularly convenient to use in my implementation.

Construct this tree using the recursive algorithm described in class maximizing the uncertainty reduction at each step. Specifically, at each non-leaf node of the tree, find the feature and threshold that maximizes the reduction in uncertainty for splitting the training set corresponding to that node with that feature and threshold. Structure your `MATLAB` code in a generic manner so any of the three uncertainty notions discussed in class can be plugged in as necessary.

Please use the following conventions to construct the tree:

- For any feature, only choose thresholds among the values of the feature in the training set.
- For any feature x_i and threshold t, let the left child correspond to the condition $x_t \leq t$, and the right child correspond to the condition $x_i > t$.
- In case there are two features x_i and x_j that both achieve the maximum reduction in uncertainty, break the tie by choosing the lower indexed one (i.e. i if $i < j$) as the feature to split on.

- If at any leaf the examples with 0 and 1 labels are equal in number, break the tie in favor of the 0 label.

A few tips for writing your code: since the construction of the tree is recursive, it's convenient to have `hw2_train_DT` call itself recursively with decreasing `depth` values. Also, at any node, for any given feature, a quick way of finding the best threshold is to sort the training data in ascending order of the feature values and compute reduction in uncertainty for thresholds equal to the feature values in the sorted list.

Write another function which computes predictions for test data given the parameters computed by the `hw2_train_DT` function, and error rate made on the test data, with the following signature:

```
function loss = hw2_test_DT(tree, test_data)
```

Part 1 of assignment: Train a decision tree using the function `hw2_train_DT` with `train_data = train_spam`, for each value of `depth` $\in \{1, 3, 6\}$ and each of the three notions of uncertainty covered in class. For every value of `depth` and each uncertainty notion, report the training error by using `hw2_test_DT` on `test_data = train_spam`, as well as the test error by using `hw2_test_DT` on `test_data = test_spam`.

Next, write a function which computes a linear classifier using the online perceptron algorithm with the following signature:

```
function weights = hw2_train_perceptron(train_data, passes)
```

The parameter `passes` is a positive integer specifying the number of passes to make over the training data. In your implementation, use the trick of adding a constant extra feature with value 1 to enable the online perceptron algorithm, which trains a homogenous linear classifier, to output a linear classifier with a non-zero threshold value. Write another function which computes predictions for test data given the weights computed by the `hw2_train_perceptron` function, and thereby computes the error rate made on the test data, with the following signature:

```
function loss = hw2_test_perceptron(weights, test_data)
```

Part 2 of assignment: Train a linear classifier using the function `hw2_train_perceptron` with `train_data = train_spam`, for each value of `passes` $\in \{1, 10, 50, 100\}$. For every value of `passes`, report the training error by using `hw2_test_perceptron` on `test_data = train_spam`, as well as the test error by using `hw2_test_perceptron` on `test_data = test_spam`.