CPSC 340 Assignment 1 (due September 23)

Data Exploration, Decision Trees, Training and Testing, Naive Bayes

- You can work on your own or in a group of 2. If you work in a group, please only hand in *one* assignment.
- Place your names and student numbers on the first page, and submit all answers as a single PDF file to handin.
- If you use information from students outside your group or from online sources, cite this at the start of each question.
- Organize your PDF file sequentially according to the section numbers in this document. Place answers (including code/figures) in the appropriate section.
- All Sections (1-4) are equally weighted.
- The code and data referred to in the assignment is available in *a1.zip*.
- Any modifications/updates to the assignment after it is first put online will be marked in red.

1 Data Exploration

Download and expand the file *a1.zip*, which contains estimates of the influenza-like illness percentage over 52 weeks on 2005-06 by Google Flu Trends. You can load this data into Matlab from the directory containing the file using:

```
load fluTrends.mat
```

This creates a matrix 'X', where each row corresponds to a week and each column corresponds to a different region (the region abbreviations are given in the 'names' variable.

1.1 Summary Statistics

Report the following statistics:

1. The minimum, maximum, mean, median, and mode of all values across the dataset.
2. The 10%, 25%, 50%, 75%, and 90% quantiles across the dataset.
3. The regions with the highest and lowest means, and the highest and lowest variances.
4. The pairs of regions with the highest and lowest correlations.

In light of parts 1 and 2, is the mode a reliable estimate of the most "common" value? Describe another way we could give a meaningful "mode" measurement for this (continuous) data.

1.2 Data Visualization

Show the following figures:

1. A plot containing the weeks on the x-axis and the percentages for each region on the y-axis.
2. A boxplot grouping data by weeks, showing the distribution across regions for each week.
3. A histogram showing the distribution of all the values in the matrix X.
4. A single histogram showing the distribution of all the columns in X.
5. A scatterplot between the two regions with lowest correlation.
6. A scatterplot between the two regions with highest correlation.

2 Decision Trees

If you run the file *example_decisionStump*, it will load a dataset containing longtitude and latitude data for 400 cities in the US, along with a class label indicating whether they were a "red" state or a "blue" state in the last election.[1] Specifically, the first column of the variable X contains the longitude and the second variable contains the latitude, while the variable y is set to 1 for blue states and 2 for red states. After it loads the data, it plots the data and then fits two simple classifiers: a classifier that always predicts the most common label (1 in this case) and a decision stump that discretizes the features (by rounding to the nearest integer) and then finds the best equality-based rule. It reports the training error with these two classifiers, then plots the decision areas made by the decision stump.

2.1 Decision Stump Implementation

Starting from the function *decisionStumpEquality*, write a new function simply called *decisionStump*. Instead of discretizing the data and using a rule based on testing an equality for a single feature, the new function should simply test for an inequality (as discussed in class). Hand in the modified function and report the updated error you obtain by using inequalities instead of discretizing and testing equality.

Note: please keep the same variable names, as subsequent parts of this assignment rely on this!

2.2 Constructing Decision Trees

Once your *decisionStump* function is finished, the function *example_decisionTree* will be able to fit a decision tree of depth 2 to the same dataset (which results in a lower training error). Look at how the decision tree is stored and how the (recursive) *predict* function works. Using the same splits as the fitted depth-2 decision tree, re-write the predict function as a simple program using if/else statements.

Modify the *depth* variable in this demo to fit deeper decision trees. Notice that the error stops decreases after a certain depth. In contrast, if you call the *decisionTreeInfoGain* you will notice that the error eventually decreases to 0. Why does the training error stop decreasing when you use classification accuracy?

[1] The cities data was sampled from `http://simplemaps.com/static/demos/resources/us-cities/cities.csv`. The election information was collected from Wikipedia.

2.3 Cost of Fitting Decision Trees

In class, we discussed how in general the decision stump minimizing the classification error can be found in $O(nd \log n)$ time. Using the greedy recursive splitting procedure, what is the total cost of fitting a decision tree of depth m in terms of n, d, and m?
(Hint: even thought there could be 2^{m-1} decision stumps, keep in mind not every stump will need to go through every example. Note also that we stop growing the decision tree if a node has no examples, so we may not even need to do anything for many of the 2^{m-1} decision stumps.)

3 Training and Testing

Notice that the *citiesSmall.mat* file also contains test data, "Xtest" and "ytest". If you run the script *example_trainTest*, after training a depth-2 decision tree (using information gain) it will evaluate the performance of the classifier on the test data. With a depth-2 decision tree, the training and test error are fairly close, so the model hasn't overfit much.

3.1 Training and Testing Error Curves

Make a plot that contains the training error and testing error as you vary the depth from 1 through 15. How do each of these errors change with the decision tree depth?

3.2 Validation Set

Suppose that we didn't have an explicit test set available. In this case, we might instead use a *validation* set. Split the training set into two equal-sized parts: use the first $n/2$ examples as a training set and the second $n/2$ examples as a validation set (we're assuming that the examples are already in a random order). What depth of decision tree would we pick if we minimized the validation set error? Does the answer change if you switch the training and validation set? How could use more of our data to estimate the depth more reliably?

4 Naive Bayes

In this section we'll first review a standard way that Bayes rule is used and then explore implementing a naive Bayes classifier, a very fast classification method that is often surprisingly accurate for text data with simple representations like bag of words.

4.1 Bayes rule for drug testing

Suppose a drug test produces a positive result with probability 0.99 for drug users, $P(T = 1|D = 1) = 0.99$. It also produces a negative result with probability 0.99 for non-drug users, $P(T = 0|D = 0) = 0.99$. The probability that a random person uses the drug is 0.001, so $P(D = 1) = 0.001$.

What is the probability that a random person who tests positive is a user, $P(D = 1|T = 1)$?

4.2 Naive Bayes by Hand

Consider the dataset below, which has 10 training examples and 2 features:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Suppose you believe that a naive Bayes model would be appropriate for this dataset, and you want to classify the following test example:

$$\hat{x} = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

(a) Compute the estimates of the class prior probabilities:

- $p(y = 1)$.
- $p(y = 0)$.

(b) Compute the estimates of the 4 conditional probabilities required by naive Bayes for this example:

- $p(x_1 = 1|y = 1)$.
- $p(x_2 = 1|y = 1)$.
- $p(x_1 = 1|y = 0)$.
- $p(x_2 = 1|y = 0)$.

(c) Under the naive Bayes model and your estimates of the above probabilities, what is the most likely label for the test example? (Show your work.)

4.3 Naive Bayes Implementation

If you run the function *example_naiveBayes* it will first load the following dataset:

1. *groupnames*: The names of four newsgroups.
2. *wordlist*: A list of words that occur in posts to these newsgroups.
3. X: A sparse binary matrix. Each row corresponds to a post, and each column corresponds to a word from the word list. A value of 1 means that the word occured in the post.
4. y: A vector with values 1 through 4, with the value corresponding to the newsgroup that the post came from.
5. *Xvalidate* and *yvalidate*: the word lists and newsgroup labels for additional newsgroup posts.

It will train a decision tree of depth 20 (this might take a while) and report its validation error, followed by training a naive Bayes model.

While the *predict* function of the naive Bayes classifier is already implemented, the calculation of the variable *p_xy* is incorrect (right now, it just sets all values to 1/2). Modify this function so that *p_xy* correctly

computes the conditional probability of these values based on the frequencies in the data set. Hand in your code and report the validation error that you obtain.

4.4 Runtime of Naive Bayes for Discrete Data

Assume you have the following setup:

- The training set has n objects each with d features.
- The test set has t objects with d features.
- Each feature can have up to c discrete values (you can assume $c \leq n$).
- There are k class labels (you can assume $k \leq n$)

You can implement the training phase of a naive Bayes classifier in this setup in $O(nd)$, since you only need to do a constant amount of work for each $X(i, j)$ value. (You do not have to actually implement it in this way for the previous question, but you should think about how this could be done). What is the cost of classifying t test examples with the model?