New Mexico State University
Klipsch School of Electrical Engineering

EE565 Pattern Recognition and Machine Learning
Fall 2016 – Project #2
Due: 5:00pm Thu. Sep. 15

Name: _____

Grade: _____

Project

The goal of this project is to build a system to classify email as either ham (good email) or spam (junk email) based on Bayesian decision theory. To begin read

`http://en.wikipedia.org/wiki/Bayesian_spam_filtering`

For this project, we will use the 2005 TREC Public Spam Corpus which consists of 92,189 labeled emails (one per file) arranged in a directory tree; ham/spam labels are in the included key.

`http://plg.uwaterloo.ca/~gvcormac/treccorpus/`

Companion files for this project may be found at

`http://www.ece.nmsu.edu/~pdeleon/Teaching/EE565/Projects/CompanionFiles2.zip`

The companion files will include a `training.idx` file which indexes the subset of email filenames to train on and the associated `training.key` which lists the corresponding labels; we will use 80% of the data for training. Also included is a `test1.idx` and `test2.idx` which indexes the subset of email filenames to test on and the associated key files; we will use 10% of the data for testing. Finally, there is an `evaluation1.idx` and `evaluation2.idx` which indexes the subset of email filenames to evaluate your system with (the key files are withheld until after the due date); we will use 10% of the data for evaluation. Your grade will not entirely be based on evaluation accuracy but this will, of course, be important. Most important, however, is your design approach, calculations, code, simulation, classifier speed, and reporting.

System Design

The project can be broken down into three codes that you will have to develop:

1. A fast and efficient parser that will scan an email and determine the unique words and count occurrences (frequencies). The parser is used for both training and test stages.

2. A training program that will gather statistical data on emails used in training. It is highly recommended that you store the data in a key-value dictionary or hash table for rapid lookup with the word as the key and the likelihoods for ham and spam as the value; in MATLAB you can use the map class, `containers.Map`. For purposes of grading, you must submit your dictionary as a text file where each row contains the information organized as follows:

   `<word>,<#occurrences in ham>,<#occurrences in spam>`

   For example, one row of `dictionary.txt` might have

   `cash,3,9`

3. A test program that will compute the posterior probabilities of the words in an email and classify an email as either ham or spam based on Bayes' theorem. The test program should read the test or evaluation index file (found in CompanionFiles2.zip), where each row is a relative path to a test email, e.g. 'data/273/092', and write an output file (like the key

files) where each row is the corresponding predicted class, i.e. ham or spam. The MATLAB script `compute_performance.m` (found in CompanionFiles2.zip) uses the input file with test filenames, output file with predicted classes, and the key for the TREC corpus to determine whether the classifications are correct or not and produces a *confusion matrix* as shown in Table 1.

Table 1: Example confusion matrix for reporting classifier results for 1000 emails and percent accuracies. The ideal classifier produces a confusion matrix with zeros in the off-diagonal elements.

		Predicted	
		ham	spam
Actual	ham	394 (90.6%)	41 (9.4%)
	spam	0 (0.0%)	565 (100.0%)

Theory

The classification process (test stage) begins by computing the *posterior* probabilities (using Bayes' theorem) for each word in the email. For example, if the first word in the email is "cash" then we have

$$p(Y = \text{ham}|X = \text{cash}) = \frac{p(X = \text{cash}|Y = \text{ham})p(Y = \text{ham})}{p(X = \text{cash})} \tag{1}$$

and

$$p(Y = \text{spam}|X = \text{cash}) = \frac{p(X = \text{cash}|Y = \text{spam})p(Y = \text{spam})}{p(X = \text{cash})}. \tag{2}$$

The *likelihoods*, $p(X|Y)$ are determined from training. You will decide the *priors*, $p(Y = \text{ham})$ and $p(Y = \text{spam})$, based on the composition of the training data or some other information. Alternatively, you may choose to not *bias* the hypothesis and choose equal priors, i.e. $p(Y = \text{ham}) = p(Y = \text{spam}) = 1/2$ in which case the *posterior* probability is equal to the *likelihood*.

Assuming the independence of the words in the email, the *posterior* probability that an email containing M words, x_m is ham is given by

$$p(Y = \text{ham}|X = \{x_1, x_2, \cdots, x_M\}) = \prod_{i=1}^{M} p(Y = \text{ham}|X = x_i) \tag{3}$$

and is spam is given by

$$p(Y = \text{spam}|X = \{x_1, x_2, \cdots, x_M\}) = \prod_{i=1}^{M} p(Y = \text{spam}|X = x_i). \tag{4}$$

The classification will be based on which posterior is maximum, i.e. maximum a posterior (MAP) decision[1] or if using equal priors, which likelihood is maximum, i.e. maximum likelihood (ML) decision.

[1]We could also normalize (3) and base the decision on whether the value is greater than 50%. For information on the normalization, see the Wikipedia entry on Bayesian spam filtering.

For better numerical behavior, you should actually use log probabilities and base the decision on the maximum of

$$\log p(Y = \text{ham}|X = \{x_1, x_2, \cdots, x_M\}) = \sum_{i=1}^{M} \log p(Y = \text{ham}|X = x_i) \tag{5}$$

or

$$\log p(Y = \text{spam}|X = \{x_1, x_2, \cdots, x_M\}) = \sum_{i=1}^{M} \log p(Y = \text{spam}|X = x_i). \tag{6}$$

Report

Please submit a 3-5 page hardcopy report describing your design approach, code, simulation, system training, and test performance given by the confusion matrices and other accuracy measures. Please `mailto:pdeleon@nmsu.edu` a zip file containing all code needed to train and test your system. Also include the data model needed for the test stage, the dictionary text file, and `evaluation1output.txt` and `evaluation2output.txt`.

Demo

Students will demonstrate their test program through the results of the two evaluations and a surprise third evaluation. A successful demonstration will produce an output file for the third evaluation with a high degree of accuracy. It goes without saying that test code should run "out-of-the-box" (no hard-coded paths) and complete in a few minutes; no training is allowed during the demonstration. As of Fall 2014, Prof. De Leon's system has 96.5% average classification accuracy and the student record is held by Rob Wauson with 98.2%.

Notes

Students are encouraged to discuss detailed, technical aspects with each other and Prof. De Leon. However, students must write all other required codes on an *individual* basis.

Hints

1. In MATLAB, the `textread` function provides flexibility comparable to that in Python for reading in a file and parsing words.
2. Prior to adding words to your dictionary, you may want to strip away punctuation, accents, and other embellishments. Depending on the complexity of your "word cleanup," you may wish to develop a function strictly for this purpose.
3. Several files in the TREC corpus are known to have virus attachments and you should skip these.