Assignment 5

CSCI2244-Randomness and Computation
Part 1 due Wednesday, March 20 at 11:59
Part 2 due Wednesday, March 27 at 11:59

The first part of the assignment consists of some routine pencil-and-paper problems about conditional probabilities and Bayes's Theorem. When you write up your solution, make sure that you carefully identify the underlying events and the conditional probabilities that are involved. (Most of these problems are lifted from the textbook by Grinstead and Snell.)

The second part of the assignment is a kind of mini-project, to create a Naïve Bayes Classifier to determine whether a text message is spam.

1 Written Problems

1. You roll a die twice. The first die comes up 4. What is the probability that the sum of the two rolls is at least 7?

2. You roll a die twice. At least one of the rolls is 4. What is the probability that the sum of the two rolls is at least 7?

3. A high school senior applies to both Boston College and Northeastern University. She estimates that the probability of her being accepted by Northeastern is 0.5, and by BC 0.3. She also estimates her probability of being accepted by both schools at 0.2. What is the probability of being admitted to Northeastern if she is also admitted to BC? Are the events 'admitted to BC' and 'admitted to Northeastern' independent events?

4. One coin in a collection of 65 coins has two heads. The rest are fair. You choose a coin at random from the collection and toss it 6 times. It comes up heads every time. What is the probability that it is the two-headed coin?

5. There are three drawers in a cabinet. One drawer contains two gold balls, one two silver balls, and the third drawer a gold and a silver ball. You pick a drawer at random, reach in and pull out a ball. It is gold. What is the probability that you picked the drawer with two gold balls?

6. *(What are the odds?)* This problem has nothing to do with conditional probability *per se,* but we need it as a warmup for the next problem. In gambling, especially, we talk about probability in terms of 'odds': We say that the odds of an event E occurring are x to y if

$$\frac{P(E)}{P(\overline{E})} = \frac{x}{y}.$$

So, for example '3 to 2 odds' means $P(E) = p$, where

$$\frac{p}{1-p} = \frac{3}{2}.$$

Solving this gives $p = \frac{3}{5}$.

Write general formulas translating odds into probabilities and vice-versa: That is, given x and y, find p such that if the odds of E occurring are x to y then $P(E) = p$. Also, given p, find x such that $P(E) = p$ implies that the odds of E occurring are x to 1.

7. We can rewrite Bayes's Theorem in terms of odds: Let us suppose we are trying to estimate the probability of an event E (for example, that a patient has condition E). The odds of E occurring in the population (*prior odds*) are $P(E)/P(\overline{E})$. After we get a test result F, the odds of E occurring (*posterior odds*) are now $P(E|F)/P(\overline{E}|F)$. Show that

$$\text{posterior odds} \;=\; \text{prior odds} \;\times L,$$

where L, the *likelihood ratio*, is defined as

$$L = \frac{P(F|E)}{P(F|\overline{E})}.$$

Now, rework the drug-testing example in the notes, using the information given in the problem to find the prior odds and the likelihood ratio, and computing the posterior odds. Your answer should agree with what was found earlier.

2 Spam Detection with a Naïve Bayes Classivier

In this problem, you will implement a Naïve Bayes Classifier to distinguish SMS spam messages.

The dataset consists of 5574 SMS messages, of which 747 have been labeled (by humans!) as spam and the other 4827 as non-spam, for which the term of art, it seems, is 'ham'. I obtained the file from the website Kaggle, but it originates from some researchers in Brazil, who describe the collection at

`http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`

The original data, in `spam2.csv`, is posted on Canvas site, so you can take a look at it. I have, however, done most of the pre-processing dirty work, and provided you with four text files,

```
spam_sms_training
ham_sms_training
spam_sms_test
ham_sms_test
```

Each of these files contains one message per line, purged of punctuation and 'stop words' like 'the', 'and', etc, and converted into lower case. The first contains half of the messages from the original dataset that were labeled as spam, the second half of the messages that were labeled as ham. The other halves are in the files with `test` in the name. These are the four files you will work with.

The goal is to use the training files to build models giving the likely distributions of words in spam and ham messages, then to use these models to classify the messages in the test files, using the Naïve Bayes method. You will submit the code you wrote, and report on how well the classifier worked.

2.1 The theory.

(You might want to read the Example below first!)

If you choose a word at random from a spam message, what is the probability that this word is, say, 'prize'? We write this as

$$P(\text{`}\texttt{prize}\text{'}|\texttt{spam}).$$

In doing so, we conceive of spam as a giant 'bag of words', in which each word appears with a particular probability, much like the urns of M& Ms described in the lecture notes. And we think of ham as another such bag.

A message D is a sequence of words

$$D = (w_1, \ldots, w_r).$$

We think of D as being generated by reaching into one of the bags r times, and pulling out the word w_i on the i^{th} draw. Thus

$$P(D|\texttt{spam}) = \prod_{i=1}^{r} P(w_i|\texttt{spam}).$$

This is the naïve part of Naïve Bayes—in effect we're treating 'you have won a prize' as just as likely to be generated as a spam message as 'prize a won have you', and ignoring anything related to the order of words or occurrence of certain multi-word phrases. While this assumption is obviously false, in practice it leads to effective results.

The task of classification is to take a message D and decide whether it is spam or not; that is, we want to decide which of the two values

$$P(\texttt{spam}|D), \quad P(\texttt{ham}|D)$$

is larger. By Bayes's Theorem,

$$P(\texttt{spam}|D) = \frac{P(D|\texttt{spam})}{P(D)} \cdot P(\texttt{spam}), \quad P(\texttt{ham}|D) = \frac{P(D|\texttt{ham})}{P(D)} \cdot P(\texttt{ham}).$$

We can ignore the common denominator $P(D)$ on the right-hand sides, and apply the naïve assumption above, so that the problem now is to determine which of the following is larger:

$$P(\texttt{spam}) \cdot \prod_{i=1}^{r} P(w_i|\texttt{spam}), \quad P(\texttt{ham}) \cdot \prod_{i=1}^{r} P(w_i|\texttt{ham}).$$

We call these two numbers the 'spam score' and the 'ham score' of the document D.

2.1.1 Estimating probabilities

To compute the scores, we need to have estimates for $P(w|\texttt{spam})$ and $P(w|\texttt{ham})$ for each word w, as well as estimates of $P(\texttt{spam})$ and $P(\texttt{ham})$. The simplest idea is to gather up all the spam messages in our training data, count the total number

$N_{\texttt{spam}}$ of words, and the total number $N_{w,\texttt{spam}}$ of occurrences of w among these words, and estimate

$$P(w|\texttt{spam}) \approx \frac{N_{w,\texttt{spam}}}{N_{\texttt{spam}}}.$$

For example, if there are a total of 6000 words in the spam messages in the training data, and 150 of them are the word 'call', then we would approximate

$$P(\text{'}\texttt{call}\text{'}|\texttt{spam}) \approx 150/6000 = 0.025.$$

Of course, we do the same thing to estimate $P(w|\texttt{ham})$.

We also need to have estimates of $P(\texttt{spam})$ and $P(\texttt{ham})$. These are our 'priors'. Here we require some kind of global estimate of the fraction of messages that are spam. For purposes of this problem, you can use the proportion of spam messages in the data set, which is about 13.4%. So we can estimate $P(\texttt{spam}) \approx 0.134$, $P(\texttt{ham}) \approx 0.866$.

2.1.2 Add-1 smoothing

What if a word, say, 'excite', appears once or twice in the ham training set, but never in the spam training set. $P(\text{'}\texttt{excite}\text{'}|\texttt{spam})$ will be then be estimated as 0, while $P(\text{'}\texttt{excite}\text{'}|\texttt{ham})$ will be positive. If we then try to evaluate a document that happens to be spam and contains this word, then the spam score will be 0, but the ham score might be non-zero, so the document will be classified incorrectly as ham. To get around this, we artificially add 1 to every count, pretending that every word encountered in the training sets appears in both ham and spam documents: Let V be the set of distinct words appearing in the two training sets, either in ham or in spam, and set for each $w \in V$,

$$P(w|\texttt{spam}) \approx \frac{N_{w,\texttt{spam}} + 1}{N_{\texttt{spam}} + |V|},$$

and likewise for ham. If the document D contains words that appear in neither training set, then we just skip these words in computing the score. In this manner, every document will get a positive spam score and a positive ham score.

2.1.3 Taking logs to avoid underflow

Typically, for any given word w, $P(w|\texttt{spam})$ and $P(w|\texttt{ham})$ will be quite small, in the neighborhood of 10^{-4} or less. If a document contains $r > 100$ words, we will wind up risking floating-point underflow when we try to evaluate the product

$$P(\texttt{spam}) \cdot \prod_{i=1}^{r} P(w_i|\texttt{spam}).$$

To avoid this, we take the logarithm of the product, which is the sum of the respective logarithms, and replace the score by

$$\ln(P(\texttt{spam})) + \sum_{i=1}^{r} \ln(P(w_i|\texttt{spam})).$$

Note that this will be a negative number, since we are taking logs of numbers less than 1. If you want, you can negate it and make the score positive, but just keep in mind that if you do this, you will want to find which of spam or ham gives a *lower* score.

2.1.4 Example

We give an example with artificially small numbers. Let's suppose we receive two packages of M&M's from two different factories, and use these as training data to construct a classifier. (Unlike the example in the notes, we don't assume any information about what the color distribution *should* be. Of course, this is a ridiculously small sample to be used for a training set.) The color counts in the two packages are as follows:

	Factory 1		**Factory 2**	
Color	Absolute	Relative	Absolute	Relative
Red	21	0.456	12	0.255
Brown	14	0.304	8	0.170
Green	6	0.130	8	0.170
Yellow	0	0	2	0.043
Blue	5	0.109	17	0.362
Total	**46**	**1.00**	**47**	**1.00**

When we apply add-1 smoothing, we modify all the absolute counts:

	Factory 1		**Factory 2**	
Color	Absolute	Relative	Absolute	Relative
Red	22	0.431	13	0.25
Brown	15	0.294	9	0.173
Green	7	0.137	7	0.173
Yellow	1	0.020	3	0.058
Blue	6	0.118	18	0.346
Total	**51**	**1.00**	**52**	**1.00**

Now suppose we receive a new bag of M&Ms whose origin is unknown, in which the color distribution is as follows:

Color	Quantity
Red	18
Brown	11
Green	10
Yellow	4
Blue	1
Orange	2
Total	**46**

This is the test data to which we apply the classifier. We believe that Factory 1 M&Ms are twice as likely to have shown up in our new shipment than Factory 2 M&Ms, so we put $P(\text{Factory 1}) = \frac{2}{3}$, and $P(\text{Factory 2}) = \frac{1}{3}$. We then compute the Factory 1 score:

$$0.431^{18} \times 0.294^{11} \times 0.137^{10} \times 0.20^{4} \times 0.118^{1} \times \frac{2}{3} \approx 10^{-29}.$$

That's an awfully small number, so we redo the calculation using logarithms:

$$18\times\ln(0.431)+11\times\ln(0.294)+10\times\ln(0.137)+4\times\ln(0.2)+\ln(0.18)+\ln\left(\frac{2}{3}\right) = -66.73.$$

Note that the orange M&M in the test data is ignored, since it does not appear in either training set.

We compute the Factory 2 score analogously to get -75.36. As $-66.73 > -75.36$, the classifier answers Factory 1.

2.2 What you should do

The training and test files have already been cleaned of punctuation and stop words, so that each line of these files consists of a single message that is simply a sequence of words. Write a function `build_models()` that opens the training files and returns two Python dictionaries, one for ham and one for spam. Each key-value pair in the dictionary should consist of a word in the combined vocabulary of the training set, and its associated smoothed probability.

Then write a function `score(message,model,prior)` that takes a cleaned message and computes its score relative to the model (which will either be the ham or spam model). The argument `prior` is the value of $P(\texttt{spam})$ or $P(\texttt{ham})$ that you use.

Finally, write a function `evaluate()` that evaluates the test data. One suggestion for how to do this–load all the ham test messages and all the spam test messages into a single list. Then repeatedly, say one thousand times, pick a message at random and use your classifier to classify it as spam or ham. How well does it work? What percentage of ham messages and of spam messages are correctly classified?