CSE 312: Foundations of Computing II
Assignment #4
January 26, 2018
due: Monday, February 5, 2018, 10:00 p.m.

Instructions:

- **Answers:** When asked for a short answer (such as a single number), also *show and explain your work* briefly. Simplify your final formula algebraically as much as possible, without using your calculator, and only then use your calculator, paying attention to the exercise's instructions for the form of your final answer. If it asks for a simplified fraction, this means an integer or a fraction a/b, where $\gcd(a, b) = 1$. If it asks for an answer to (say) 3 significant digits, this means that, when written in scientific notation, your answer would look like $a.bc \times 10^d$, where a, b, c are digits and d is an integer. Highlight or box your final answer to each exercise part. If you are using indicator random variables, be sure that you *define them explicitly.*
- **Turn-in:** You will submit a program source file for Exercise #6 via a Google Form (details below). You will upload a single PDF file containing your solutions to all the remaining exercises on Gradescope. Do not write your name on your pages (your Gradescope account will identify you to us) and do not include a copy of the exercise's question in what you turn in. Each numbered homework question must be answered on its own page (or pages). You must follow the Gradescope prompts that have you link exercise numbers to your pages. You may typeset your solutions on a computer or you can handwrite them, take a picture of (or scan) each handwritten page, and convert the pictures into a single PDF file. You are responsible for making sure that your solution is easily readable. Popular choices for typesetting mathematics are Microsoft Word (be sure to use its Equation Editor and save as PDF) and LaTeX.

1. Let X be the number of heads seen when 3 coins are flipped independently. The first coin has probability p of heads, the second has probability q of heads, and the third has probability r of heads.

   (a) Calculate the probability mass function for X as a function of p, q, r.

   (b) Find $E[X]$ directly by applying the definition of expectation to the result from part (a). Simplify your answer as far as possible, which will be important for comparing your answer to (c) below.

   (c) Find $E[X]$ again by using linearity of expectation.

   Look back at your work and think about which way of computing the expectation was simpler, (a) + (b), or (c).

2. Suppose that you want a fair coin toss, but the only coin C you have has probability p of heads, where the value of p is unknown to you. (This problem has computing applications: you might want your computer to have access to a true random number generator, but your source of randomness is something like a Geiger counter detecting alpha particles.) The Hungarian mathematician John von Neumann devised a brilliant but simple method for doing this. Toss the biased coin C twice: (1) if the outcome is HT, treat it as heads, (2) if the outcome is TH, treat it as tails, and (3) if the outcome is HH or TT, repeat the procedure.

   (a) Calculate the expected number of tosses of C required to generate a single fair coin toss, as a function of p. Then evaluate this expected number for $p = 1/10$, giving an exact value as a simplified fraction.

   (b) Calculate the expected number of tosses of C required to generate k fair coin tosses, as a function of p and k. Be sure to justify your answer.

3. I have played 27735 deals against the devilish Schnapsen program *Doktor Schnaps*. I am convinced that I am never dealt enough trumps due to my bad luck.

   (a) In a single deal, what is the probability of being dealt no trumps in your initial 5-card hand? Give your answer to 4 significant digits. Check your answer against the lecture notes where I solved this problem to 3 significant digits, to be sure yours is consistent.

   (b) Define a "run of bad luck" to be 3 deals in a row in which you are dealt no trumps in your initial hand in each of the 3 deals. These runs are allowed to overlap so, for instance, 6 deals in a row can contribute up to 4 runs of bad luck of 3 deals each. In 27735 independent deals of Schnapsen, what is the expected number of runs of bad luck? Give your answer to 4 significant digits.

4. You have an array holding 5 numbers, and you run the following algorithm to find the maximum of those numbers:

```
max = a[0];
for(i = 1; i < 5; i++){
    if(max < a[i]){  // (*)
        max = a[i];  // (**)
    }
}
```

   To do a detailed analysis of the running time of this algorithm, you would need to know how many times statement (**) is skipped. A related, perhaps simpler, question (and the only one you have to answer in this exercise) is this: assuming that the 5 numbers are all distinct and equally likely to be in any order (that is, they might as well be a random permutation of 1, 2, 3, 4, 5), let the random variable X be the number of times the "if" test in (*) evaluates to false before the first time statement (**) is executed (if ever). For example, $X = 2$ means that the 0-th array element was at least as great as elements 1 and 2, but less than element 3, and $X = 4$ means that the 0-th element was at least as great as each of the others. Give exact answers as simplified fractions.

   (a) Find the probability mass function for X.

   (b) Find $E[X]$.

5. A certain bubble gum company includes a picture card of a famous computer scientist in each pack of bubble gum it sells. Needless to say, kids absolutely love collecting these exciting cards. A complete set of cards consists of n different pictures. Suppose that every pack you buy is equally likely to contain any of the n pictures, independent of the other packs.

   (a) Bob buys a pack a day until he has a complete set. Let random variable Y be the number of packs required to accomplish this objective. What is $\mathrm{E}[Y]$? (Hint: Define random variable Y_i as the number of packs Bob buys from the time he first has $i - 1$ different pictures until the first time he acquires one that he doesn't have yet.)

   The n-th *harmonic number* is defined as $H_n = 1 + 1/2 + 1/3 + 1/4 + \cdots + 1/n$. As a function of n, H_n grows very similarly to $\ln n$. In fact, $H_n = \ln n + \Theta(1)$. Give your answer as a function of H_n.

   (b) For $n = 75$, calculate $\mathrm{E}[Y]$ and round it to the nearest integer. You may use any calculator you like for this task.

6. Use the Naive Bayes Classifier to implement a spam filter that learns word spam probabilities from our pre-labeled training data and then predicts the label (ham or spam) of a set of emails that it hasn't seen before. **You may use Java or Python3.** We've provided starter code in Java and Python.

   Download

`http://courses.cs.washington.edu/courses/cse312/18wi/312B/homework/NaiveBayes.zip`

and unzip it. Inside the "data" folder, the emails are separated into "train" and "test" data. Each "train" email is already labeled as either *spam* or *ham*, and they should be used to train your model and word probabilities. The "test" data is not labeled, and they are the emails whose labels you will use your classifier to predict.

The emails we are using are a subset of the Enron Corpus, which is a set of real emails from employees at an energy company. The emails have a subject line and a body, both of which are "tokenized" so that each unique word or bit of punctuation is separated by a space or newline. The provided starter code takes a filename and returns a set of all of the distinct tokens in the file.

Some requirements and advice for the implementation:

- You will likely want to iterate over all the files in a directory. `os.listdir()` and `File.listFiles()` will be useful in Python and Java, respectively.
- Your program must accept the file path to your data directory as a command line argument, for example,

  ```
  $ java NaiveBayes /path/to/your/data/directory
  ```

  The only paths hardcoded in your program should be to subdirectories of the directory given as the command line input and should be specified relative to that command line input ("./train/ham/", "./train/spam/", "./test/"). If you are using Windows, change all occurrences of "\" to "/" in these hardcoded paths before turning in your program.
- Read about how to avoid floating point underflow in the notes.
- Do not use integer division when generating your word probabilities.
- Think about the data structures you want to use to keep track of the word counts and probabilities.
- **If you use Eclipse, remove all package statements before you turn in your source code.**

For your output, for each file in the test data, print the filename, a space, and either the word ham or spam depending on how your program classified that file. **Print to stdout, and not to a file.** See `example_output.txt` for an example of the format. Note that we may run your code on test data you haven't seen yet.

After you've classified the 500 test emails, you can compare your results with the actual labels that we hid from you, by using the output checker here. It is not expected that Naive Bayes will classify every single test email correctly, but you should probably aim to get at least 90-95% of them classified correctly, and certainly do better than random chance. We are not grading you on whether you classify 100% of the examples accurately, but rather on general program correctness.

Needless to say, you should practice what you've learned in other courses: document your program, use good variable names, keep your code clean and straightforward, etc. Include comments outlining what your program does and how. We will not spend time trying to decipher obscure, contorted code.

What to turn in: Turn in your source file at `https://goo.gl/forms/52zw4QekHIIUWA8s1`. To do so, you must be logged into your uw.edu account in Google. Modify naive_bayes.py, or NaiveBayes.java, if you use the starter code. If you don't, please use the same file names to make our grading scripts happy. Do not turn in the training/test data that we provide you. *Be careful not to turn in our unmodified starter code instead of your solution code by mistake.* You can submit multiple times: the latest submission will overwrite previous submissions.