CSE 312: Foundations of Computing II Autumn 2023

Problem Set 3

Due: Wednesday, October 18, by 11:59pm

Instructions

Solutions format, collaboration policy, and late policy. See PSet 1 for further details. The same requirements and policies still apply. Also follow the typesetting instructions from the prior PSets.

Solutions submission. You must submit your solution via Gradescope. In particular:

- For the solutions to Task 1-7 (and Task 9 if you choose to do it), submit under "PSet 3 [Written]" a *single* PDF file containing the solution to all tasks in the homework. Each numbered task should be solved on its own page (or pages). Follow the prompt on Gradescope to link tasks to your pages. Do not write your name on the individual pages – Gradescope will handle that.
- For the programming part (Task 8), submit your code under "PSet 3 [Coding]" as a file called $\mathrm{nb.py}$.

Task 1 – Balls [10 pts]

Consider an urn containing 15 balls, of which 9 are white and the rest are black. A sample of size 4 is to be drawn **(a)** with replacement, and **(b)** without replacement. What is the conditional probability (in each case) that the first and third balls drawn will be white given that the sample drawn contains exactly 3 white balls?

Note that drawing balls *with replacement* means that after a ball is drawn (uniformly at random from the balls in the bin) it is put back into the urn before the next independent draw. If the balls are drawn *without replacement*, the ball drawn at each step (uniformly at random from the balls in the bin) is not put back into the urn before the next draw.

Please use the following notation in your answer: Let W_i be the event that the i^{th} ball drawn is white. Let B_i be the event that that the i^{th} ball drawn is black, and let F be the event that exactly 3 white balls are drawn.

Task 2 – Doggone, Doggtwo, Doggthree... [10 pts]

A hunter has two hunting dogs. One day, on the trail of some animal, the hunter comes to a place where the road diverges into two paths. She knows that each dog, independent of the other, will choose the correct path with probability p. The hunter decides to let each dog choose a path, and if they agree, take that one, and if they disagree, to randomly pick a path. What is the probability that she ends up taking the correct path?

Hint: Use the law of total probability, partitioning based on whether the dogs choose the same path or different paths.

Task 3 – Jokers [12 pts]

Suppose that we have a modified deck of cards that consists of the ordinary deck of 52 cards plus 4 jokers. This deck of 56 cards is randomly divided into 4 hands of 14 cards each. We are interested in determining p, the probability that each hand has a joker. Let E_i be the event that the i-th hand has exactly one joker. Use the **chain rule** to determine

$$p = \mathbb{P}(E_1 \cap E_2 \cap E_3 \cap E_4)$$

Task 4 – The Random Programmer [9 pts]

You need to include a certain subroutine in your code for your job. There is a commercial software that has this subroutine, but it is not open-source, so you decide to write the subroutine yourself. However, you want to test the subroutine you write for correctness before deploying it. (You may assume that the commercially available software is correct.)

To perform this test, you compare the outputs of your code with those of the commercial software for the same inputs. Call your subroutine Y and the commercial software's subroutine C; so, you want to test if $Y(x) = C(x)$. *Assume the inputs are all integers.*

Suppose that, based on the subroutine you have written, you know for certain that if your subroutine is incorrect then it will be wrong almost all the time: there will be at most d integers (call these $r_1, r_2, \ldots, r_d$) at which $Y(r_i) = C(r_i)$ holds while *for all* $x \notin \{r_1, \ldots, r_d\}$, $Y(x) \neq C(x)$. Unfortunately, you don't have any idea what $r_1, \ldots, r_d$ are!

With this problem setup, let us devise a good testing strategy.

a) Suppose, to test your code, you choose an integer in the range $\{1, 2, 3, \ldots, 11\,d\}$ uniformly at random. What is the **maximum probability** (corresponding to worst possible values of the unknown $r_1, \ldots, r_d$) that you mistakenly deduce that your code is correct, even when it's not? (From here on, we term such a situation "a failure of the testing strategy".)

b) You want to improve the probability that the testing strategy does *not* fail. One way to do this, based on the preceding answer, is to increase the range of integers you test over. What is the **maximum probability** of failure of the testing strategy if you increase the range of integers to $\{1, 2, 3, \ldots, 1000000\,d\}$?

c) In practice, the range of values you can use is limited by the machine-precision available, so this isn't a sustainable process. You therefore dial back the range of values to the original $\{1, 2, 3, \ldots, 11\,d\}$ and turn to an alternate approach. Assume that you have an error in your subroutine and you repeat the test strategy k times, each trial independent of the other. Assume that you are "sampling with replacement"; i.e., in *each* of the k trials, you have the *entire* range of $\{1, 2, 3, \ldots, 11\,d\}$ to pick values from. In this case, what is the maximum probability that your testing strategy fails?

Task 5 – Conditional probability that you knew all along [10 pts]

You are taking a multiple choice test that has 5 answer choices for each question. In answering a question on this test, the probability you know the correct answer is p. If you don't know the correct answer, you choose one (uniformly) at random. What is the probability that you knew the correct answer to a question, given that you answered it correctly?

Task 6 – Are you game? [12 pts]

The Octopus Game Show has its contestants compete in various dangerous and/or embarrassing tasks. Based on whether they succeed in a week's task they are randomly chosen to go on to the next week. The Octopus Game Show randomly chooses some of those to continue on to the next week with different probabilities based on whether or not they succeeded in the immediately prior week. (Because the organizers think it is fun to watch people fail, success does not guarantee moving on, and failure doesn't guarantee being eliminated from the game.)

Suppose that the Octopus Game Show sets the probabilities for these selections each week based on the fraction of successful contestants in the prior week so that a random contestant will be advanced with probability 50%. Suppose that you also know that the probabilities of selection are such that in expectation:

- 84% of those who make it to the second week were successful in the first week.
- 72% of those who *do not* make it to the second week were successful in the first week.

We randomly choose a contestant uniformly from among those who started the game.

a) What is the probability that this contestant was successful in the first week?

b) Expressed as a percentage with 2 decimal places, what is the probability that the Octopus Game Show selected the contestant for the second week conditioned on their being successful in the first week?

c) Expressed as a percentage with 2 decimal places, what is the probability that the contestant was not selected for the second week conditioned on their being unsuccessful in their first week?

Task 7 – RV or Motor Home? [12 pts]

Let Ω be the sample space consisting all possible selections of 3 distinct elements from $[5] = \{1, 2, 3, 4, 5\}$. Assume a uniform distribution on Ω. Define random variable X on Ω by $X(\omega) = \min\{i \mid i \in \omega\}$ for $\omega \in \Omega$.

a) What is $X(\Omega)$?

b) Give explicit simplified values in fractional or decimal form for the probability mass function p_X.

c) Compute $\mathbb{E}[X]$.

Task 8 – Naive Bayes [Coding] [25 pts]

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. See the slides from Section 3 for details on implementation. To solve the task, we have set up an edstem lesson. In particular, write your code to implement the functions fit and $\mathrm{predict}$ in the provided file, $\mathrm{nb.py}$.

You will be able to run your code directly within edstem, and to test it, using the "Mark" option. This, however, will not evaluate your solution. Instead, once you're ready to submit, you can right-click the

files in the directory to download them. Please upload your completed $\mathrm{nb.py}$ to Gradescope under "PSet3 [Coding]".

Some notes and advice:

- Read about how to avoid floating point underflow using the log-trick in the notes.
- Make sure you understand how Laplace smoothing works.
- Remember to remove any debug statements that you are printing to the output.
- **Do not directly manipulate file paths or use hardcoded file paths.** A file path you have hardcoded into your program that works on your computer won't work on the computer we use to test your program.
- 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. Your score on Gradescope is your final score, as you have unlimited attempts. **START EARLY**.
- We will evaluate your code on data you don't have access to, in addition to the data you are given.
- Remember, it is not expected that Naive Bayes will classify every single test email correctly, but it should certainly do better than random chance! As this algorithm is deterministic, you should get a certain specific test accuracy around 90-95%, which we will be testing for to ensure your algorithm is correct. Note that we will run your code on a test dataset you haven't seen, but you will know immediately if you got full score.

Task 9 – Extra Credit: Foibles on an airplane [15 pts]

There are n passengers boarding an airplane that has exactly n seats. Everyone has a ticket with an assigned seat number, corresponding to the passenger number itself, i.e. passenger i has seat number i for $i \in [n]$.

However, the first passenger has lost their ticket and takes a random seat. Every subsequent passenger after the first does one of two things

1. If their assigned seat is not taken, they sit in that seat.
2. If their assigned seat is taken, they take a random seat (with their choice being uniformly random over the empty seats remaining).

Suppose that you are the very last passenger to board the plane. What is the probability that you will get your assigned seat?

Note: This is a really cool problem that illustrates some of the recurring simplicity we see in probability where problems sound complicated but the answer turns out to be surprisingly elegant and intuitive!