CSE 312: Foundations of Computing II — Summer 2022

Problem Set 5 (due Friday, July 29, 11:59pm)

Directions: *For each problem, explain/justify how you obtained your answer, as correct answers without an explanation may receive **no credit**. Moreover, in the event of an incorrect answer, we can still try to give you partial credit based on the explanation you provide. Unless you are asked to, you should leave your answer in terms of factorials, combinations, etc., for instance* 26^7 *or* $26!/7!$ *or* $26 \cdot \binom{26}{7}$.

***Submission**: You must upload a **pdf** of your solutions to Gradescope under "Pset 5 [Written]". The use of LaTex is highly recommended, and we have provided a template. (Note that if you want to hand-write your solutions, you'll need to scan them. If we cannot make out your writing, your work may be ungradable, so make sure it is legible.) Your code will be submitted as a .py file under "Pset 5 [Coding]".*

Instructions as to how to upload your solutions to Gradescope are on the course web page.

*Remember that you must tag your written problems on Gradescope, or you will potentially receive **no credit** as mentioned in the syllabus. Please put each numbered problem on its own page in the pdf (this will make selecting pages easier when you submit), and ensure that your pdfs are oriented correctly (e.g. not upside-down or sideways). As stated above, the coding problem will also be submitted to Gradescope.*

***Collaboration**: This pset must be submitted **individually**. You are welcome and encouraged to discuss approaches with your fellow students, but everyone **must write up their own solutions.** Failure to do so is an instance of academic dishonesty.*

1. Random Grades (20 points)

Every week, 20,000 students flip a 10,000-sided fair dice, numbered 1 to 10,000, to see if they can get their GPA changed to a 4.0. If they roll a 1, they win (they get their GPA changed). You may assume each student's roll is independent. Let X be the number of students who win.

(a) [4 Points] For any given week, give the appropriate probability distribution (including parameter(s)), and find the expected number of students who win.

(b) [8 Points] For any given week, find the exact probability that at least 2 students win. Give your answer to 5 decimal places.

(c) [8 Points] For any given week, estimate the probability that at least 2 students win, using the Poisson approximation. Give your answer to 5 decimal places.

2. Instagram (25 points)

A photo-sharing startup offers the following service. A client may upload any number N of photos and the server will compare each of the $\binom{N}{2}$ pairs of photos with their proprietary image matching algorithms to see if there is any person that is in both pictures. Testing shows that the matching algorithm is the slowest part of the service, taking about 100 milliseconds of CPU time per photo pair. Hence, estimating the total time spent processing per client upload is a big part of sizing their data center. You (an engineer) say, "N is a random variable, and so we have to estimate the time using probability". What will the **expected** time (in milliseconds) for CPU demand per client be (as a function of p or λ) if N follows...

(a) [7 Points] the Poisson distribution with parameter λ?

(b) [7 Points] the geometric distribution with parameter p?

(c) [11 Points] $N = 80X + 5$, where X is a Bernoulli random variable with parameter p?

In each case, include as part of your answer the expected value of N and the variance of N. Make sure your answer is **not** in the form of a summation for this problem.

Hint: Be careful about what linearity of expectation allows us to do! It's not magic, it has specific limitations.

3. Exponential Darts (15 points)

You throw a dart at a circular target of radius r. Let X be the distance of your dart's hit from the center of the target. You have improved and your aim is such that $X \sim \text{Exponential}(4/r)$. (Note that it is possible for the dart to completely miss the target.)

(a) [8 Points] As a function of r, determine the value m such that $\Pr(X < m) = \Pr(X > m)$. Then, for $r = 10$, give the value of m to 3 decimal places.

(b) [7 Points] What is the probability that you miss the target completely? Give your answer to 3 decimal places.

4. The Classic Flea Problem (20 points)

A flea of negligible size is trapped in a large, spherical, inflated beach ball with radius r. At this moment, it is equally likely to be at any point within the ball. Let X be the distance of the flea from the center of the ball. For X, find

(a) [6 Points] the cumulative distribution function F.

(b) [4 Points] the probability density function f.

(c) [4 Points] the expected value.

(d) [6 Points] the variance.

Reminder: the volume of a sphere of radius r is $\frac{4}{3}\pi r^3$.

5. Explore the zoo! (20 points)

[**Coding**] Understanding the process that leads to different random variables is a great way to gain familiarity for what they mean. For each random variable, write a function that simulates its generation process. Your function should return a random sample of that rv, with the appropriate probability. The **only** function you **can and should** use to generate randomness is np.random.rand(): a function that returns a uniform random float in the range $[0, 1]$. Note that a function from one part may call a function from a previous part if you wish. For more clarity, we are asking you to generate a random sample from a particular distribution; multiple calls to your function can and should return different values in its range, approximately matching that variable's probability mass function. Also note that you don't need to know how to use Negative Binomial and Hypergeometric random variables, but just how to generate them as code.

Write your code for the following parts in the provided file: `cse312_pset5_gen_rvs.py`

(a) $X \sim Ber(p)$: 1 with probability p and 0 with probability $1 - p$.

(b) $X \sim Bin(n, p)$: the number of heads in n independent flips of a coin with probability of heads p. Implement the function `gen_bin`.

(c) $X \sim Geo(p)$: the number of flips up to and including the first head, when the probability of heads is p. Implement the function `gen_geo`.

(d) $X \sim NegBin(r, p)$: the number of flips up to and including the r-th head in independent coin tosses, when the probability of heads is p. Implement the function `gen_negbin`.

(e) $X \sim HypGeo(N, K, n)$: the number of kit kats you get when you grab n random candies from a bag consisting of N total candies, only K of which are kit kats. Implement the function `gen_hypgeo`.

(f) $X \sim Poi(\lambda)$: the number of events in a minute, where the historical rate is λ events per minute. Implement the function `gen_poi`.

(g) Given an arbitrary list (or numpy array) of probabilities, like $\vec{p} = [0.1, 0.3, 0.4, 0.2]$, sample an index with the appropriate probability. That is, return 0 with probability 0.1, 1 with probability 0.3, 2 with probability 0.4, and 3 with probability 0.2. Implement the function `gen_arb`.