CSE 312: Foundations of Computing II Autumn 2022

Problem Set 6

Due: Wednesday, November 16, 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 1-5, submit under "PSet 6 [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 6), submit your code under "PSet 6 [Coding]" as one file, $\mathrm{min\_hash.py}$.

Task 1 – Normal, normal, normal [15 pts]

Your answers should be correct to four decimal places.

a) Apparently IQ is roughly normally distributed, with a mean of 100 and a standard deviation of about 15. What fraction of people would be classified as "genius" if that means IQ of 140 or above? (These numbers are made up.)

b) The height of a group of people is approximately normally distributed with a mean of 175 cm and a standard deviation of 7.5 cm.

- Approximately what fraction of these people are 155 cm tall or less?
- If we form a basketball team by choosing 5 people from this group uniformly and independently at random, calculate the probability that at least one of them is over 190 cm tall.

Task 2 – CLT in "real life" [10 pts]

Error-correcting codes[1] are used in order to compensate for errors in transmission of messages (and in recovery of stored data from unreliable hardware). You are on a mission to Mars and need to send regular updates to mission control. Most of the packets actually don't get through, but you are using an error-correcting code that can let mission control recover the original message you send so as long as at least 128 packets are received (not erased). Suppose that each packet gets erased independently with probability 0.6. How many packets should you send such that you can recover the message with probability at least 99%

Use the Central Limit Theorem to approximate the answer, using the continuity correction. Final answer should be an integer number of packets and your intermediate calculations should be correct to sufficient precision (e.g. 4 decimal places) to ensure a good approximation.

[1] From the Wikipedia page: "In computing, telecommunication, information theory, and coding theory, an *error correcting code* (ECC) is used for controlling errors in data over unreliable or noisy communication channels. The central idea is that the sender encodes the message with redundant information in the form of an ECC. The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission."

Task 3 – Statistics Books [25 pts]

Alice is going shopping for statistics books for H hours, where H is a random variable, equally likely to be 1, 2 or 3. The number of books B she buys is random and depends on how long she is in the store for. We are told that

$$\mathbb{P}(B = b \mid H = h) = \frac{1}{h}, \qquad \text{for } b = 1, \ldots, h\ .$$

a) Find the joint distribution of B and H using the chain rule.

b) Find the marginal distribution of B.

c) Find the conditional distribution of H given that $B = 1$ (i.e., $\mathbb{P}(H = h \mid B = 1)$ for each possible h in 1,2,3). Use the definition of conditional probability and the results from previous parts.

d) Suppose that we are told that Alice bought either 1 or 2 books. Find the expected number of hours she shopped conditioned on this event. Use the definition of conditional expectation and Bayes Theorem.

Task 4 – Duck Hunt [15 pts]

Ten hunters are waiting for ducks to fly by. A flock of ducks flies overhead with the number of ducks in the flock a Poisson random variable with mean 5. Suppose that each hunter chooses one of the ducks to aim at uniformly at random and independent of the choices of the other hunters. The hunters all fire at the same time. If each hunter independently hits their chosen target with probability 0.3, use the law of total expectation to compute the expected number of ducks that are hit.

Task 5 – Joint Densities [15 pts]

Suppose that X, Y are jointly continuous rv's with joint density

$$f_{X,Y}(x, y) = \begin{cases} 6x^2 y & 0 \leqslant x \leqslant 1, 0 \leqslant y \leqslant 1 \\ 0 & \text{otherwise} \end{cases}$$

(Observe that this is a probability density function since it is non-negative and we can use nested integrals to show that

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(xy)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1\int_0^1 6x^2 y\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 \left(3x^2y^2\Big|_{y=0}^{y=1}\right)\mathrm{d}x = \int_0^1 3x^2\,\mathrm{d}x = x^3\Big|_{x=0}^{x=1} = 1.)$$

Your answers below should **not** be evaluated unless otherwise specified. Your answers should usually be in terms of integrals or nested double integrals.

a) Write an expression used nested integrals that we can evaluate to find $\mathbb{P}(Y \geqslant X)$. Hint: draw the region of the joint density, and the desired region.

b) Write an expression using using a single integral that we can evaluate to find the marginal density $f_X(x)$. Be sure to specify the value of $f_X(x)$ for all $x \in \mathbb{R}$. Do the same for $f_Y(y)$.

c) Are X and Y independent? Justify your answer. (You may need to evaluate an integral or two to do this.)

Task 6 – Distinct Elements [20 pts]

Recall the setup for the MinHash algorithm presented in class. The universe of is the set $\mathcal{U}$ (think of this as the set of all 8-byte integers), and we have a single **uniform** hash function $h : \mathcal{U} \to [0, 1]$. That is, for an integer y, pretend $h(y)$ is a **continuous** $\mathrm{Unif}(0, 1)$ random variable. That is, $h(x_1), h(x_2), ..., h(x_N)$ for any N **distinct** elements are iid continuous $\mathrm{Unif}(0, 1)$ random variables, but since the hash function always gives the same output for some given input, if, for example, the i-th user ID x_i and the j-th user ID x_j are the same, then $h(x_i) = h(x_j)$ (i.e., they are the "same" $\mathrm{Unif}(0, 1)$ random variable).

Then, the MinHash algorithm is realized by the following pseudocode, which explains its two key functions:

1. UPDATE(x): How to update your variable when you see a new stream element.
2. ESTIMATE(): At any given time, how to estimate the number of distinct elements you've seen so far.

Note that this differs from the syntax used on the slides, but captures the same algorithm.

MinHash Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow \min\{\text{val}, h(x)\}$
function ESTIMATE() **return** round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▷ Loop through all stream elements
  UPDATE(x_i) ▷ Update our single float variable
return ESTIMATE() ▷ An estimate for n, the number of distinct elements.

To help you out with the following questions, we have set up an edstem lesson. However, you are required to upload your final solution to Gradescope (see instructions above).

a) Implement the functions UPDATE and ESTIMATE in the MinHash class of min_hash.py.

b) The estimator we used in **a)** has high variance, and therefore it may not always give good answer. As outlined in class, we improve this by considering k variables

$$\text{val}_1, \text{val}_2, \ldots, \text{val}_k$$

where each val_i, $1 \leqslant i \leqslant k$ is an i.i.d. random variable with the distribution of the minimum of $m \leqslant N$ independent $\mathrm{Unif}(0, 1)$ variables, obtained by hashing the N elements in the stream with independent hash functions $h^1, \ldots, h^k$. Our final estimate will then be

$$\hat{n} = \frac{1}{\widehat{\text{val}}} - 1 \quad \text{where} \quad \widehat{\text{val}} = \frac{1}{k} \sum_{i=1}^{k} \text{val}_i.$$

Implement the functions UPDATE and ESTIMATE in the MultMinHash class of min_hash.py using the improved estimator.

Refer to Section 9.5 of the book for more details on the distinct elements algorithm.