CSE 312: Foundations of Computing II Autumn 2023

Problem Set 6

Due: Wednesday, November 15, 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 1-4, 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 5), submit your code under "PSet 6 [Coding]" as one file, min_hash.py.

Task 1 – Kangaroos [15 pts]

The average leap of a kangaroo is 20 feet. However, because of various factors such as strength, wind, etc, the kangaroo doesn't always leap exactly 20 feet. A zoologist tells you that the kangaroo leap is normally distributed with mean 20 and variance 9.
In all parts below, give your answer to 5 decimal places.

a) (5 points) What is the probability that the kangaroo leaps more than 25 feet?

b) (5 points) If the kangaroo leaps 3 times, what is the probability that at least one of the leaps is more than 25 feet? All the kangaroo's leaps are independent and identically distributed.

c) (5 points) What is the probability that the kangaroo leaps between 13 and 27 feet?

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 0.7. 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 – Sticks [20 pts]

Consider a stick of length 1. We break the stick at a position sampled uniformly along its length and throw away the shorter part. We use the random variable X to represent the length of the part we keep and random variable $Z \in [0,1]$ to denote where we broke it. We do this a second time on the piece of stick we kept, breaking it at a random point along its length, and keeping the longer part. We use the random variable Y to denote the length of the part we keep after the second break.

a) Let $x \in \mathbb{R}$. For which values of Z is $X \leqslant x$?

b) Compute $F_X(x) = P(X \leqslant x)$ using what you showed in part (a).

c) Compute f_X from F_X. What distribution from the Zoo is f_X?

d) Conditioned on $X = x$, what is the distribution of Y? (Hint: Everything here is a re-scaling of what went on in the first round. You don't need to rederive everything again.)

Task 4 – Joint Densities [15 pts]

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

$$f_{X,Y}(x,y) = \begin{cases} 10x^4 y & 0 \leqslant x \leqslant 1, 0 \leqslant y \leqslant 1 \\ 0 & \text{otherwise} \end{cases}$$

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

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(xy)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1\int_0^1 10x^4 y\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 \left(5x^4y^2\Big|_{y=0}^{y=1}\right)\mathrm{d}x = \int_0^1 5x^4\,\mathrm{d}x = x^5\Big|_{x=0}^{x=1} = 1.)$$

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 5 – Distinct Elements [Coding] [20 pts]

Recall the setup for the $\mathrm{MinHash}$ 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 $h : \mathcal{U} \rightarrow [0, 1]$. 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 $\mathrm{MinHash}$ algorithm is realized by the following pseudocode, which explains its two key functions:

1. $\mathrm{UPDATE}(x)$: How to update your variable when you see a new stream element.
2. $\mathrm{ESTIMATE}()$: 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 ← $\infty$
function UPDATE($x$)
    val ← $\min\{\text{val}, h(x)\}$
function ESTIMATE() return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \dots, N$: 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 $\mathrm{ESTIMATE}$ in the $\mathrm{MinHash}$ class of $\mathrm{min\_hash.py}$.

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

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

where each of 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 $h^1, \dots, h^k$. 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 $\mathrm{ESTIMATE}$ in the $\mathrm{MultMinHash}$ class of $\mathrm{min\_hash.py}$ using the improved estimator.

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