COMPSCI 514: Problem Set 2

Due: 10/15 by 11:59pm in Gradescope.

Instructions:

- You are allowed to, and highly encouraged to, work on this problem set in a group of up to three members.
- Each group should **submit a single solution set**: one member should upload a pdf to Gradescope, marking the other members as part of their group in Gradescope.
- You may talk to members of other groups at a high level about the problems but **not work through the solutions in detail together**.
- You must show your work/derive any answers as part of the solutions to receive full credit.

1. Random Group Testing (10 points)

Suppose that we wish to control the spread of some contagious illness by testing as many individuals as possible. Unfortunately testing is expensive. A common method to address is issue is to test individuals in *groups*. I.e., the biological samples from multiple patients are combined into a single sample and tested for the disease all at once. If the test returns negative, it means that all individuals in the group are negative. If the test comes back positive, it means that at least one individual in the group has the disease. We'll see how this strategy can be used to significantly save on the number of tests required to identify a small subset of positive individuals.

1. (4 points) Show that there is a deterministic algorithm that uses $O(\sqrt{nk})$ tests and, if at most k individuals in the populations are positive, correctly identifies all these individuals, without identifying any individuals that are negative. You may assume that k is known in advance (often it can be estimated from the positive rate of prior tests). **Hint:** Consider a two-stage approach that first tests groups and then tests individuals within the groups.
2. (1 points) Say we are testing the UMass Amherst student body. $n = 30,000$ and there is a 1% positivity rate, so $k = 300$. How many tests does the strategy of part (1) save over simply individually testing each member of the student body? You may assume that the constant in the big-Oh notation is 2 – i.e. that the algorithms uses $\leq 2\sqrt{nk}$ tests. There is an algorithm achieving that constant.
3. (3 points) Consider the following randomized scheme: collect r samples from each individual. Then, repeat the following process r times: randomly partition the population into G groups (i.e., each individual is assigned independently to group i with probability $1/G$ for $i = 1, 2, ..., G$), and test each group in aggregate. Once this process is complete, report that an individual is positive if *every group they were part of tested positive*. Report that an individual

is negative if *any of the groups they were part of tested negative.* Show that for $G = O(k)$ this scheme finds all truly positive patients, and that each negative patient is marked positive with probability $\leq \frac{1}{2^r}$.

4. (2 points) Show that if we set $r = O(\log n)$, then with probability $\geq 99/100$ the method of part (3) yields no false positives, no false negatives, and requires just $O(k \log n)$ tests.

2. Missing Analysis for Distinct Elements Algorithms (5 points)

For a continuous random variable X, we define the *probability density function* as $f_X(t) = \frac{d}{dt} \Pr[X \leq t]$. Then $\mathbb{E}[X] = \int_{-\infty}^{\infty} t f_X(t) dt$ and $\mathbb{E}[X^2] = \int_{-\infty}^{\infty} t^2 f_X(t) dt$. Suppose we pick d uniform random values in the range $[0, 1]$ independently and X is defined to be the smallest of these values.

1. (2 points) What is the value of $\mathbb{E}[X]$?
2. (2 points) What is the value of $\mathbb{E}[X^2]$?
3. (1 point) What is the value of $\mathrm{Var}[X]$?

Hint: You may use Wolfram Alpha (or similar) to evaluate any integrals you encounter. Note that we claimed bounds on these quantities in lectures but did not prove them and that's what we're are doing here.

3. Distinct Elements Revisited (6 points)

Suppose we have a rough estimate d' of the number of distinct values in a stream $x_1, \ldots, x_m$. Specifically, suppose $d/2 \leq d' \leq 2d$ where d is the exact value. However, we want to find an estimate d'' such that $|d'' - d| \leq \epsilon d$ where ϵ is an arbitrarily small positive value. Let $h : U \to [d']$ be a fully random hash function and let X be the number of distinct values in the stream that evaluate to 1 when the hash function is applied.

1. (2 points) Compute the expected value of X and conclude that if you can approximate $\mathbb{E}[X]$ up to error $\epsilon\mathbb{E}[X]$ then you learn d up to error ϵd.
2. (2 points) Analyze how you can use multiple hash functions to learn $\mathbb{E}[X]$ up to error $\epsilon\mathbb{E}[X]$ with probability at least $1 - \delta$.
3. (2 points) If you implemented this approach as a data stream algorithm, what is the expected space use of the algorithm?

4. Join Size Estimation via Hashing (9 points)

One common application of random hashing is in database systems. Consider estimating the *inner join size* of two tables without performing an actual expensive inner join. This is useful, e.g., in database query optimization – approximate join sizes can be used to chose an efficient execution for a complex query involving multiple joins.

Formally, consider two sets of keys $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ which are subsets of some universe U. Our goal is to estimate $|A \cap B|$ based on compact representations of A and B. Consider compressing the sets as follows:

- Choose k independent uniform random hash functions $\mathbf{h}_1, \ldots, \mathbf{h}_k : U \to [0, 1]$.
- Let $\mathbf{s}^A = [\mathbf{s}_1^A, \ldots, \mathbf{s}_k^A]$ where $\mathbf{s}_i^A = \min_{j=1,\ldots,m} \mathbf{h}_i(a_j)$.

- Let $\mathbf{s}^B = [\mathbf{s}^B_1, \ldots, \mathbf{s}^B_k]$ where $\mathbf{s}^B_i = \min_{j=1,\ldots,n} \mathbf{h}_i(b_j)$.

Given $\mathbf{s}^A$ and $\mathbf{s}^B$, each a list of k numbers, we estimate join size $|A \cap B|$ as $Z = \mathbf{c} \cdot (\frac{1}{\mathbf{s}} - 1)$ where $\mathbf{c} = \frac{1}{k}\sum_{i=1}^k \mathbb{1}[\mathbf{s}^A_i = \mathbf{s}^B_i]$ (i.e., $\mathbf{c}$ is the fraction of colliding hashes in $\mathbf{s}^A$ and $\mathbf{s}^B$) and

$$\mathbf{s} = \frac{1}{k}\sum_{i=1}^k \min(\mathbf{s}^A_i, \mathbf{s}^B_i).$$

1. (3 points) Show that if we set $k \geq \frac{1}{\epsilon^2\delta}$, for $\epsilon, \delta \in (0,1)$, then with probability at least $1-\delta$, $|\mathbf{c} - J(A,B)| \leq \epsilon\sqrt{J(A,B)}$.

2. (3 points) Show that if we set $k = O(\frac{1}{\epsilon^2\delta})$, for $\epsilon, \delta \in (0,1)$, then with probability at least $1-\delta$, $\left|\left(\frac{1}{\mathbf{s}} - 1\right) - |A \cup B|\right| \leq \epsilon|A \cup B|$.

3. (3 points) Conclude that if we set $k = O(\frac{1}{\epsilon^2\delta})$, for $\epsilon, \delta \in (0,1)$ then with probability at least $1-\delta$,

$$|Z - |A \cap B|| \leq \epsilon\sqrt{|A \cap B||A \cup B|}.$$

5. Estimating Sum of Squares (6 points)

Let $x_1, x_2, \ldots x_m \in [n]$ be a stream and define $f_i = |\{j \in [m] : x_j = i\}|$, i.e., f_i is the number of elements in the stream that equal i. In the question, we are interested in estimating the sum of squared frequencies $F_2 = \sum_i f_i^2$. Consider the following data stream algorithm that uses $O(\log m)$ bits of space to store a single counter:

1. Let $h : [n] \to \{-1, 1\}$ be a fully random hash function (i.e., for all x, $h((x) = 1$ with probability $1/2$, $h(x) = -1$ with probability $1/2$, and for all $x \neq y$, $h(x)$ and $h(y)$ are independent.)

2. $C \leftarrow 0$

3. For $i = 1$ to m: $C \leftarrow h(x_i) + C$

4. Return C^2

You will first analyse the expected value and variance of the above algorithm and then use this to design an algorithm for estimating F_2.

1. (2 points) What is the expected value of the output?

2. (2 points) Prove that the variance of the output is $O(F_2^2)$.

3. (2 points) Use the above algorithm as a building block towards a small-space streaming algorithm for estimating F_2 up to a factor $1+\epsilon$ with probability at least $1-\delta$. The algorithm should use $O(\epsilon^{-2}\log(1/\delta)\log m)$ bits of space.

6. Locality Sensitive Hashing in Use (10 points)

We would like to use locality sensitive hashing to search for similar handwritten digit images from the MNIST dataset. We will measure similarity using cosine similarity and use the SimHash method. Throughout the problem, use the data provided in the `mnist.mat` file: `https://people.cs.umass.edu/~cmusco/CS514F21/psets/mnist.mat`. It is helpful to initially normalize all images to have unit Euclidean norm. Include printouts of any code in your problem set submission.

Given an input image x, you would like to identify any image y with cosine similarity $\langle x, y\rangle \geq .95$. Your task is to pick a number of table repetitions t and a hash signature length r so that any image close to x is identified with probability at least 98%. At the same time, you would like to minimize the number of false positives in your hashing scheme.

1. (2 points) Using that for two images x and y, $\Pr[SimHash(x) = SimHash(y)] = 1 - \frac{\theta}{\pi}$ where θ is the angle between x and y in radians, determine for each $r \in \{1, \ldots, 30\}$ the number of repetitions t required to achieve the desired false negative rate of 2%. You may want to write code to solve this problem, but please also describe in words/equations how you determined the required t for a given r. Assume that x and y are unit norm through this problem.

2. (2 points) Given a fixed value of r and t, what is the expected number of collisions between images x and y with cosine similarity $\langle x, y\rangle = s$ across t hash tables? Give an equation in terms of r, t, and s. Count collisions happening in different tables as different collisions. Assume that a collision only occurs if x and y have matching SimHash signatures (i.e., ignore additional collisions that occur when inserting to the hash table).

3. (2 points) For each $r \in \{1, \ldots, 30\}$, use the $10,000$ images in `testX` to estimate the expected number of collisions that will be encountered for an image x when using enough hash tables to ensure a 2% false negative rate (as determined in part (1)). Plot the expected number of collisions as a function of r. Discuss the trend, and how you might use this information to choose r for an application. Include both a description with words/equations along with any code used. **Hint:** For a given image x in `testX`, compute the expected total number of collisions that will occur with other images in `testX`. Then average over all $10,000$ possibilities for x in the set to get your estimate.

4. (4 points): To get a feel for how SimHash is working, set $r = 35$ and compute SimHash signatures for the $60,000$ images in `trainX`. For simplicity, do not worry about using any repetitions (i.e., use $t = 1$.) Focusing on a single digit type, run a few near neighbor queries in order to find a few (maybe 6 or so) sets of images that hash to the same signatures. Plot these sets of colliding images. What do you notice? Are the colliding images similar? Are there many false positives?