COMPSCI 514: Problem Set 2

Due: 10/16 by 11:59pm in Gradescope.

Instructions:

- You are allowed 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 should separately submit the core competency problems from any challenge problems you choose to complete. These do not necessarily need to be submitted with the same groups.
- 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.

Core Competency Problems

1. Ways to Bound Maximum Server Loads (8 points)

Suppose there are n requests and k servers. Each request is equally likely to be assigned to any of the servers and all requests are assigned independently of the others. Let $\mathbf{R}_i$ be the number of requests assigned to the ith server.

1. (2 points) Prove that for any $z \leq n$, $\Pr[\mathbf{R}_1 \geq z] \leq \binom{n}{z} \cdot 1/k^z$. **Hint:** Consider the union bound with events of the form E_A where A is a subset of $\{1, \ldots, n\}$.
2. (2 points) Prove that for any $z \leq n$, $\Pr[\max(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_k) \geq z] \leq n^z/k^{z-1}$. **Hint:** Note that $\binom{a}{b} \leq a^b$ for any positives integers a and b where $b \leq a$.
3. (2 points) If $n = k/2$, prove that all servers receive $\leq 2\log_2 k$ requests with probability at least $1 - 1/k$. Here $\log_2 k$ is base 2.
4. (2 points) If $n = k/2$ use a Chernoff bound to prove that all servers receive $\leq 4\log_2 k$ requests with probability at least $1 - 1/k$. **Hint:** You may assume that $k \geq 2$. You may be able to get a much tighter bound than what we are asking for – we are allowing slack so that you can make simplifications to your calculations and still obtain the desired bound.

2. Machine Learning and Bloom Filters (8 points)

Consider applying Bloom filters to database optimization: you work for an online content provider and want to store a large table of $(user, video)$ pairs. Any time a user watches a video, you store information about when they watched it, how they rated it, etc. Most $(user, video)$ pairs have no information stored in the table since your platform serves a large number of videos and any user has only watched a tiny fraction of them. So, whenever a user watches a video, you insert information

on that pair into the database and also add the pair to a bloom filter. Before you make a possibly expensive query to the database to look up information about a $(user, video)$ pair, you check the bloom filter and only query the database if it returns a hit.

1. (2 points) You have $s = 1$ billion $(user, video)$ pairs with information stored in your database. You would like to store these pairs in a bloom filter with false positive rate of 10%. How large must your filter be to achieve this rate? I.e., how large must you set the bit array size m and how much storage in kBs, MBs, or GBs does it take? You may use the false positive rate calculation given in class of $\left(1 - e^{\frac{-kn}{m}}\right)^k$ – even though its classic derivation is incorrect, it is a good approximation. Throughout you should assume that you use the optimal number of hash functions $k = \ln 2 \cdot \frac{m}{n}$ and for simplicity, that this value is an integer.

2. (2 points) The machine learning team at your company has developed a classifier that is able to predict if a user has watched a video with very good accuracy, without having to look at the user-video database. The classifier is a function $y : (user, video) \to \{0, 1\}$ which returns 1 on 95% of $(user, video)$ pairs where the user has actually watched the video and returns 0 on 95% of pairs where the user hasn't watched the video.

   You decide to use this classifier to reduce the load on your bloom filter. You only add a $(user, video)$ pair (u, v) to the filter if the user has watched the video *and* $y(u, v) = 0$. When any pair (u, v) is queried you first check if $y(u, v) = 1$ – if so you query the pair in the your database. If not, you check your bloom filter, and if it returns a positive you then query the database.

   What is the overall false positive rate on a random pair (u, v) where u has not watched v? Give an expression in terms of the number of pairs with information stored in the database s, the filter size m, and the number of hash functions k.

   Hint: Think about how many items actually get inserted into your bloom filter in this modified approach. It might be helpful to draw a flow-chart corresponding to the algorithm described above.

3. (2 points) How large must you set m to still have a false positive rate of 10%. How much storage have you saved using this ML approach compared to the standard approach in (1).

4. (2 points) Consider the following simple approach which uses no Bloom filter at all: you only check a (u, v) pair in the database when $y(u, v) = 1$. If $y(u, v) = 0$, you do not check the pair in the database. Why might you not want to use this approach?

3. Boosting the Success of Randomized Algorithms (4 points)

1. (4 points) Suppose I have a randomized algorithm that solves a decision problem (i.e., a problem with a 'yes' or 'no' answer) with worst case runtime T and at least $2/3$ probability of correctness on any input. For any $\delta \in (0, 1)$, describe and analyze a randomized algorithm that correctly answers the same problem with probability at least $1 - \delta$ and has worst case runtime $O(\log(1/\delta) \cdot T)$. **Hint:** Use a similar idea to the median trick.

4. Set Similarities and Locality Sensitive Hash Functions (6 points)

You would like to use shingling and locality sensitive hashing to identify possible plagiarism in student essays.

1. (2 points) Consider an essay A with 1000 unique shingles in it and a publication B with 3000 unique shingles, let S_A and S_B be the shingle sets for A and B respectively. So $|S_A| = 1000$ and $|S_B| = 3000$. Assuming the essay A was fully copied from a portion of B, what is the Jaccard similarity between these sets? What is $\Pr(MinHash(S_A) = MinHash(S_B))$?

2. (2 points) Consider another essay C also with 1000 unique shingles that is not copied and only shares 100 shingles with B. Compute $\Pr(MinHash(S_C) = MinHash(S_B))$.

3. (2 points) Find a signature length r and repetition parameter t such that the fully copied essay A is identified with LSH-based similarity search with probability $\geq .95$ and the non-copied essay C is identified with probability $\leq .05$. Focus on minimizing the space complexity (i.e., the number of hash tables t used). By 'identified', we mean that the essay falls in the same bucket as B in at least one of the t hash tables.

   Hint: It may be helpful (although is not required) to write a very simple program to help solve this one. Assume for simplicity, as in class that there are no 'spurious collisions'. That is, if two items have different hash signatures, they will never be mapped to the same bucket of the hash table.

Challenge Problems

C1. Revisiting Mark-and-Recapture for Wikipedia 🌶🌶🌶

In Challenge Problem 2 of the first problem set, we attempted to use mark-and-recapture to estimate the number of articles on English language Wikipedia. However, we obtained consistent underestimates for the article count. We suspected that this was due to the fact that the random article feature doesn't return truly uniformly random articles (see https://en.wikipedia.org/wiki/Wikipedia:FAQ/Technical#random). Here, we will analyze carefully how the distribution of the random article generated effects our results.

We will model the random article generator as follows: there are n articles. Each is independently assigned a uniformly random real number $r_i \in [0, 1]$, which serves as its id. To randomly sample an article, we sample a uniformly random real number $x \in [0, 1]$. We then return the article with the lowest id greater than x. If there is such no article (i.e., all articles have ids $< x$), then we return the article with the lowest id.

Hint: It maybe be helpful to visualize the process by considering a circle with circumference 1. The articles are each placed uniformly at random at n points along this circle according to their ids. An article is returned by the random article generator if the random point x lies in between that article and the article directly adjacent to it, counter-clockwise.

1. Let $\mathbf{p}_i$ denote the probability that article i is returned by the generator. Note that $\mathbf{p}_i$ is a random variable which depends on $r_1, \ldots, r_n$. What is $\mathbb{E}[\mathbf{p}_i]$? **Hint:** Use linearity of expectation and symmetry.

2. For any t, what is $\Pr[\mathbf{p}_i \geq t]$? **Hint:** What sequence of $n-1$ events needs to happen for the event $\mathbf{p}_i \geq t$ to occur?

3. What is $\mathbb{E}[\mathbf{p}_i^2]$? What is $\mathbb{E}[\sum_{i=1}^n \mathbf{p}_i^2]$? **Hint:** Use the approach here as we applied to min-hash in class: for a non-negative random variable $\mathbf{X}$, $\mathbb{E}[\mathbf{X}] = \int_0^\infty \Pr[\mathbf{X} \geq t]\, dt$.

4. Use part (3) to explain why our Wikipedia article count estimates were roughly half of what we expected in Problem Set 1. **Hint:** You may want to look at Problems 3 and C1 from Problem Set 1 for guidance.

C2. A Faster Algorithm for Distinct Elements

Let $\mathbf{x}_1, \ldots, \mathbf{x}_d$ be chosen independent and uniformly at random in $[0, 1]$. For any $k \in \{1, 2, \ldots, d-1\}$, let $\mathbf{s}$ be the $(k+1)^{st}$ smallest value of $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$. Let $\mathbf{s}_1$ be the k^{th} smallest of $\{\mathbf{x}_2, \ldots, \mathbf{x}_d\}$. I.e., $\mathbf{s}_1$ is the k^{th} smallest of all the values except the first one. Finally, let $\mathcal{S} \subset \{1, \ldots, d\}$ be the set $\mathcal{S} = \{i \in \{1, \ldots, d\} | \mathbf{x}_i < \mathbf{s}\}$. That is, $\mathcal{S}$ contains the indices of the k values lying below $\mathbf{s}$.

1. Let $\mathbf{Y} = \begin{cases} 1/\mathbf{s} \text{ if } 1 \in \mathcal{S} \\ 0 \text{ otherwise} \end{cases}$. Let $\mathbf{Y}_1 = \begin{cases} 1/\mathbf{s}_1 \text{ if } 1 \in \mathcal{S} \\ 0 \text{ otherwise} \end{cases}$. Argue that $\mathbf{Y} = \mathbf{Y}_1$.
2. Prove that $\mathbb{E}[\mathbf{Y}] = 1$. **Hint:** First consider $\mathbb{E}[\mathbf{Y}_1 | \mathbf{s}_1 = t]$ for any $t \geq 0$.
3. Prove that $\mathbb{E}[\mathbf{Y}] = \mathbb{E}\left[\frac{1}{\mathbf{s}}\right] \cdot \frac{k}{d}$. Conclude that $\mathbb{E}\left[\frac{1}{\mathbf{s}}\right] = \frac{d}{k}$.
4. Use a similar approach to show that $\text{Var}[\frac{1}{s}] = \frac{d(d-k)}{k^2(k-1)}$.
5. Argue that, for any $\epsilon, \delta \in (0, 1)$, if we set $k \geq \frac{1}{\epsilon^2 \delta} + 1$ and $\tilde{d} = \frac{k}{\mathbf{s}}$, then with probability at least $1 - \delta$, we will have $|\tilde{d} - d| \leq \epsilon d$.
6. Describe how the above analysis can be used to obtain an algorithm for distinct items estimation using $O(1/(\epsilon^2 \delta))$ units of space (i.e., matching the algorithm discussed in class) but just one hash function. Why might this algorithm be preferable to the one discussed in class?

C3. Exponential Tail Bounds from Scratch

Here we will see how to prove exponential tail bounds from scratch, using the moment generating function approach discussed in class. We will prove a generalization of the Chernoff bound to random variables lying in $[0, 1]$, rather than just $\{0, 1\}$. Throughout, let $\exp(x)$ denote e^x.

1. Let $\mathbf{Y}$ be a random variable that takes values in the interval $[0, 1]$. Prove that for $t > 0$, $\mathbb{E}[\exp(t\mathbf{Y})] \leq 1 + \mathbb{E}[\mathbf{Y}](e^t - 1)$. **Hint:** First show that $\exp(ty) \leq 1 + y(e^t - 1)$ for all $y \in [0, 1]$. It might be helpful to plot these two functions to compare them before trying to give a formal proof.
2. Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be independent random variables that take values in the interval $[0, 1]$. Let $\mathbf{X} = \sum_{i \in [n]} \mathbf{X}_i$ and $\mu = \mathbb{E}[\mathbf{X}]$. Prove that for $0 < \delta < 1$,

$$\Pr[\mathbf{X} \geq (1+\delta)\mu] = \Pr[\exp(t\mathbf{X}) \geq \exp(t(1+\delta)\mu)] \leq \frac{\mathbb{E}[\exp(t\mathbf{X})]}{\exp(t(1+\delta)\mu)} .$$

   Hint: Use Markov's inequality.
3. Prove that $\Pr[\mathbf{X} \geq (1+\delta)\mu] \leq \exp(-\delta^2 \mu / 3)$.

   Hint: Consider setting $t = \ln(1+\delta)$ and using part one of the question. You may use the fact that $(1+\delta)^{1+\delta} \geq e^{\delta + \delta^2/3}$ and that for any $x \geq 0$, $(1+x) \leq e^x$. You may also want to recall that for independent random variables $\mathbf{Y}, \mathbf{Z}$, $\mathbb{E}[\mathbf{YZ}] = \mathbb{E}[\mathbf{Y}] \cdot \mathbb{E}[\mathbf{Z}]$.

   Hint: For simplicity, you may assume here that each $\mathbf{X}_i$ has the same mean. I.e., that $\mathbb{E}[\mathbf{X}_i] = \mu/n$. The stated bound holds even without this assumption, but making this assumption may make the proof easier, and will suffice for achieving full credit.