New York University Tandon School of Engineering
Computer Science and Engineering

# CS-GY 6763: Homework 2.
Due Thursday, February 20th, 2025, 11:59pm ET.

*Collaboration is allowed on this problem set, but solutions must be written-up individually. Please list collaborators for each problem separately, or write "No Collaborators" if you worked alone.*

## Problem 1: Exploring Concentration Bounds

**(8 pts)** Let $X$ be a random variable uniformly distributed in the interval $[0, 1]$. Since we know $X$'s distribution exacty, we can easily check that $\Pr[X \geq 7/8] = 1/8$. But let's take a look at what various concentration inequalities would predict about this probability using less information about $X$.

1. Given an upper bound on $\Pr[X \geq 7/8]$ using Markov's inequality.
2. Give an upper bound on $\Pr[X \geq 7/8]$ using Chebyshev's inequality.
3. Given an upper bound on $\Pr[X \geq 7/8]$ by applying Markov's inequality to the random variables $X^2$ (the "raw" second moment). Note that this is slightly different than using Chebyshev's inequality, which applies Markov to the "central" second moment $(X - \mathbb{E}[X])^2$.
4. What happens for higher moments? Applying Markov's to $X^q$ for $q = 3, 4, \ldots, 10$. Describe what you see in a sentence. What value of $q$ gives the tightest bound?
5. One take-away here is that, depending on the random variable being studied, it is not always optimal to use the variance as a deviation measure to show concentration. Markov's can be used with any monotonic function $g$, and as we see above, different choices might give better bounds. Exhibit a monotonic function $g$ so that applying Markov's to $g(X)$ gives as tight an upper bound on $\Pr[X \geq 7/8]$ as you can. Maximum points if you can get $\Pr[X \geq 7/8] \leq 1/8$, which would be the best possible.

## Problem 2: Boosting Success Probability

**(12 pts)** We saw in class that computing the *mean* of many different answers from a randomized algorithm is often an effective way to improve accuracy. It turns out that the *median* is also very useful: it can be used to improve *success probability*.

In particular, suppose our goal is to estimate some scalar quantity $f(X)$ of a dataset $X$ (e.g, the number of distinct items in a stream). Suppose we have developed a high-accuracy randomized algorithm, $\mathcal{A}$, but it only succeeds with probability $\geq 2/3$. I.e., on any input $X$, with with probability $\geq 2/3$,

$$|\mathcal{A}[X] - f(X)| \leq \epsilon$$

Prove that, if we run $\mathcal{A}$ a total of $r = O(\log(1/\delta)$ times with different random seeds and return the *median*, $M$, of the $r$ outputs, then with probability $1 - \delta$:

$$|M - f(X)| \leq \epsilon.$$

This approach could be used, for example, to obtain an $O(\log(1/\delta)/\epsilon^2)$ space algorithm for distinct items, whereas in class we only proved $O(1/\delta\epsilon^2)$ using Chebyshev's inequality.

**Hint:** Think about what event would have to happen for the median to be "bad", i.e., to give error $> \epsilon$. If the median is bad, what does it say about the fraction of our $r$ estimates that are bad?

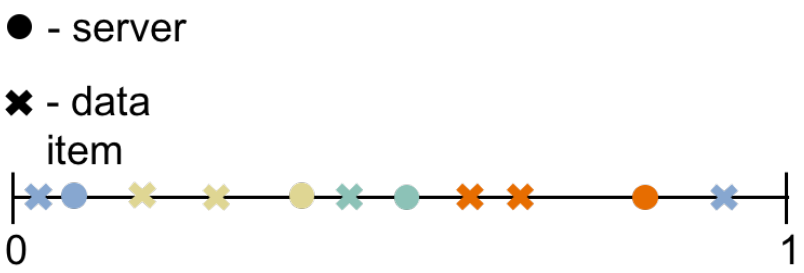


Figure 1: Each data item is stored on the server with matching color.

## Problem 3: Hashing around the clock.

**(15 pts)** In modern systems, hashing is often used to distribute data items or computational tasks to a collection of servers. What happens when a server is added or removed from a system? Most hash functions, including those discussed in class, are tailored to the number of servers, $n$, and would change completely if $n$ changes. This would require rehashing and moving all of our $m$ data items, an expensive operation.

Here we consider an approach to avoid this problem. Assume we have access to a completely random hash function that maps any value $x$ to a real value $h(x) \in [0, 1]$. Use the hash function to map *both* data items and servers randomly to $[0, 1]$. Each data item is stored on the first server to its right on the number line (with wrap around – i.e. a job hashed below 1 but above all serves is assigned to the first server after 0). When a new server is added to the system, we hash it to $[0, 1]$ and move data items accordingly.

1. Suppose we have $n$ servers initially. When a new server is added to the system, what is the expected number of data items that need to be relocated?

2. Show that, with probability $> 9/10$, no server "owns" more than an $O(\log n/n)$ fraction of the interval $[0, 1]$. **Hint:** This can be proven without a concentration bound.

3. Show that if we have $n$ servers and $m$ items and $m > n$, the maximum load on any server is no more than $O(\frac{m}{n} \log n)$ with probability $> 9/10$.

## Problem 4(a): Analyzing Sign-JL and JL for Inner Products

**(15 pts)** Often practitioners prefer JL matrices with discrete random entries instead of Gaussians because they take less space to store and are easier to generate. We analyze one construction below.

Suppose that $\mathbf{\Pi}$ is a "sign Johnson-Lindenstrauss matrix" with $n$ columns, $k$ rows, and i.i.d. $\pm 1$ entries scaled by $1/\sqrt{k}$. In other words, each entry in the matrix has values $-1/\sqrt{k}$ with probability $1/2$ and $1/\sqrt{k}$ with probability $1/2$.

1. Prove that for any vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbb{E}[\|\mathbf{\Pi x}\|_2^2] = \|\mathbf{x}\|_2^2$ and that $\text{Var}[\|\mathbf{\Pi x}\|_2^2] \leq \frac{2}{k}\|\mathbf{x}\|_2^4$. This is the meat of the problem and will take some effort.

2. Use the above to prove that $\Pr\left[\left|\|\mathbf{\Pi x}\|_2^2 - \|\mathbf{x}\|_2^2\right| \geq \epsilon\|\mathbf{x}\|_2^2\right] \leq \delta$ as long as we choose $k = O\left(\frac{1/\delta}{\epsilon^2}\right)$. Note that this bound almost matches the distributed JL lemma proven in class, but with a worse failure probability dependence of $1/\delta$ in place of $\log(1/\delta)$.

   *With more work, it's possible to improve the dependence to* $\log(1/\delta)$ *for the sign-JL matrix, but we won't do so here.*

3. Generalize your analysis above to show that JL matrices are also useful in approximating inner products between two vectors. For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ prove that $\Pr\left[|\langle \mathbf{\Pi x}, \mathbf{\Pi y}\rangle - \langle \mathbf{x}, \mathbf{y}\rangle| \geq \epsilon\|\mathbf{x}\|_2\|\mathbf{y}\|_2\right] \leq \delta$ as long as we choose $k = O\left(\frac{1/\delta}{\epsilon^2}\right)$.

   *This result can also be improved to have a* $\log(1/\delta)$ *dependence in place of* $1/\delta$. .

## Problem 4(b): Join Size Estimation

**(5 pts)** One powerful application of sketching is in database applications. For example, a common goal is to estimate the *inner join size* of two tables without performing an actual inner join (which is expensive, as it requires enumerating the keys of the tables). Formally, consider two sets of unique keys $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ which are subsets of $1, 2, \ldots, U$. Our goal is to estimate $|X \cap Y|$ based on small space compressions of $X$ and $Y$.

Using your result from Problem 1, describe a method based on inner product estimation that constructs independent sketches of $X$ and $Y$ of size $k = O\left(\frac{1}{\epsilon^2}\right)$ and from these sketches can return an estimate $Z$ for $|X \cap Y|$ satisfying

$$|Z - |X \cap Y|| \leq \epsilon \sqrt{|X||Y|}$$

with probability 9/10.