New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 9223I: Homework 1.
Due Thursday, September 26th, 2019, 11:59pm.

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.

For just this first problem set, 10% *extra credit will be given if solutions are typewritten (using LaTeX, Markdown, or some other mathematical formatting program).*

Problem 1: Short answers.

(10 pts) Do these first!

1. For any given k, give an example of a random variable for which Chebyshev's inequality is tight up to constant factors. Specifically, for any given k, describe a random variable X with variance σ^2 such that $\Pr[|X - \mathbb{E}X| \geq k\sigma] \geq \frac{1}{10k^2}$.
2. A biased random coin comes up heads with probability $1/n$ for some $n > 1$. Show that, after n random flips, the probability that you never see heads is $\leq .3679$. Show that after $n \log n$ flips, the probability that you never see heads is $\leq 1/n$. **Hint:** Think back to calculus and the definition of e!
3. Suppose that Π is a Johnson-Lindenstrauss matrix with $O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ rows. Prove that for any x, y:

$$|\langle x, y\rangle - \langle \Pi x, \Pi y\rangle| \leq \epsilon(\|x\|_2^2 + \|y\|_2^2)$$

with probability $\geq 1 - \delta$.

Problem 2: 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? For most hash functions, including those discussed in class, the hash function is 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.

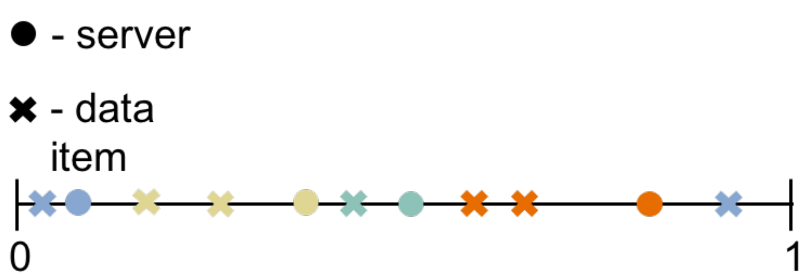

Figure 1: Each data item is stored on the server with matching color.

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 $[0, 1]$.

3. Show that if we have n servers and m items and $m > n$, the maximum load on any server is $O(\frac{m}{n}\log(n))$ with probability $> 9/10$.

Problem 3: Pinning down the median.

(15 pts) A very common objective in statistical analysis is to estimate the *median* (not the mean) of a dataset from uniformly random samples. For example, a census might poll random citizens in a city to request information about their income. From this sample, the goal is to estimate the city's *median income*.

1. Suppose we have a list S of n numbers with median M. We sample k numbers $X_1, \ldots, X_k$ uniformly at random (with replacement) from S. Show that as long as $k \geq O\left(\frac{\log 1/\delta}{\epsilon^2}\right)$, then $\tilde{M} = \text{median}(X_1, \ldots, X_k)$ is a good approximate median in the following sense: with probability $(1 - \delta)$, at least a $\frac{1}{2} - \epsilon$ fraction of numbers in S are $\leq \tilde{M}$ and at least a $\frac{1}{2} - \epsilon$ fraction of numbers in S are $\geq \tilde{M}$.

2. **Extra Credit – optional!** Show that it is *impossible* to estimate the *value* of the true median M with $o(n)$ random samples from S, even if we just want to get within a constant approximation factor, and succeed with constant probability. For example, we can't even guarantee that $.5M \leq \tilde{M} \leq 2M$ with probability $\geq 2/3$ unless we take nearly n samples from S.

Problem 4: Compressed classification.

(10 pts) In machine learning, the goal of many classification methods (like support vector machines) is to separate data into classes using a *separating hyperplane*.

Recall that a hyperplane in $\mathbb{R}^d$ is defined by a unit vector $a \in \mathbb{R}^d$ ($\|a\|_2 = 1$) and scalar $c \in \mathbb{R}$. It contains all $h \in \mathbb{R}^d$ such that $\langle a, h \rangle = c$.

Suppose our dataset consists of n unit vectors in $\mathbb{R}^d$ (i.e. each data point is normalized to have norm 1). These points can be separated into two sets X, Y, with the guarantee that there exists a hyperplane such that every point in X is on one side and every point in Y is on the other. In other words, for all $x \in X, \langle a, x \rangle > c$ and for all $y \in Y, \langle a, y \rangle < c$.

Furthermore, suppose that the ℓ_2 distance of each point in X and Y to this separating hyperplane is at least ϵ. When this is the case, the hyperplane is said to have "margin" ϵ.

1. Show that this margin assumption equivalently implies that for all $x \in X, \langle a, x \rangle > c + \epsilon$ and for all $y \in Y, \langle a, y \rangle < c - \epsilon$.

2. Show that if we use a Johnson-Lindenstrauss map Π to reduce our data points to $O(\log n/\epsilon^2)$ dimensions, then the dimension reduced data can still be separated by a hyperplane with margin $\epsilon/4$, with high probability (say 99/100 times).