NYU TANDON SCHOOL OF ENGINEERING

New York University Tandon School of Engineering
Computer Science and Engineering

# CS-GY 6763: Homework 2.
Due Tuesday, October 18th, 2022, 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: Analyzing Sign-JL and JL for Inner Products

**(20 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 $\mathrm{Var}[\|\mathbf{\Pi x}\|_2^2] \leq \frac{2}{k}\|\mathbf{x}\|_2^4$. 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 2: Join Size Estimation

**(15 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 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 $Y$. We consider two approaches below.

1. 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 $9/10$.

2. Alternatively, consider compressing the sets as follows:

   - Choose $k$ uniform random hash functions $h_1, \ldots, h_k : \{1, \ldots, U\} \to [0, 1]$.
   - Let $C^X = [C_1^X, \ldots, C_k^X]$ where $C_i^X = \min_{j=1,\ldots,m} h_i(x_j)$.
   - Let $C^Y = [C_1^Y, \ldots, C_k^Y]$ where $C_i^Y = \min_{j=1,\ldots,n} h_i(y_j)$.

Given the sketches $C^X$ and $C^Y$., which each contain $k$ numbers, we estimate join size as $Z = \frac{k'}{k} \cdot (\frac{1}{S} - 1)$ where $k' \leq k$ equals $k' = \sum_{i=1}^{k} \mathbb{1}[C_i^X = C_i^Y]$ and

$$S = \frac{1}{k}\sum_{i=1}^{k} \min(C_i^X, C_i^Y).$$

Show that if we set $k = O(\frac{1}{\epsilon^2})$ then with probability $9/10$,

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

**Hint:** Think about the two pieces of the estimator $Z$, $k'/k$ and $(\frac{1}{S} - 1)$, separately. What quantities do we expect these random variables to be close to?

3. Which method give better accuracy? The JL based method or the hashing based method?

## Problem 3: Concentration of sum of random vectors.

**(10 pts)** We have seen that several concentration inequalities apply to sums of *bounded* random variables (Hoeffding, Chernoff, etc.). In this problem you will prove a basic concentration result for sums of *bounded* random *vectors*. Let $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^d$ be $d$ dimensional i.i.d. random vectors (independent, drawn from the same distribution) with mean $\boldsymbol{\mu}$ – i.e. $\mathbb{E}[\mathbf{x}_i] = \mu$. Further suppose that, deterministically, $\|\mathbf{x}_i\|_2^2 \leq C$ for some fixed constant $C$. Let $\mathbf{s} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i$. Prove that if $k \geq O(\frac{1/\delta}{\epsilon^2})$ then

$$\Pr\left[\|\mathbf{s} - \boldsymbol{\mu}\|_2 \geq \epsilon\sqrt{C}\right] \leq \delta.$$

*Try solving the problem first under the assumption that $\boldsymbol{\mu} = \mathbf{0}$, then reduce the general problem to the mean $\mathbf{0}$ case.*

**COMPLETE EITHER PROBLEM 4 OR PROBLEM 5**

## 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 $\ell_2$ distance of each point in $X$ and $Y$ to this separating hyperplane is at least $\epsilon$. When this is the case, the hyperplane is said to have "margin" $\epsilon$.

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

2. Show that if we use a Johnson-Lindenstrauss map $\Pi$ 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 $> 9/10$).

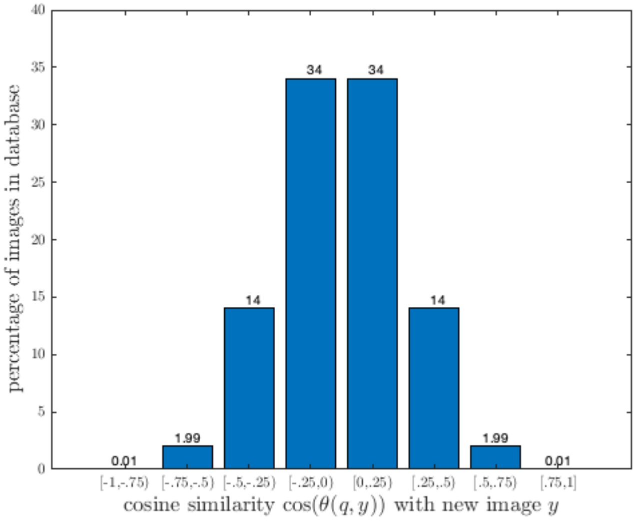
## Problem 5: LSH in the Wild

*This exercise does not involve formal proofs or analysis like more typical problem set problems. It will likely involve some coding or spreadsheet work.*

**(10 pts)** To support its largely visual platform, Pinterest runs a massive image de-duplication operation built on Locality Sensitive Hashing for Cosine Similarity. You can read about the actual system here. All information and numbers below are otherwise purely hypothetical.

Pinterest has a database of $N = \mathbf{1}$ **billion** images. Each image in the database is pre-processed and represented as a vector $\mathbf{q} \in \mathbb{R}^d$. When a new image is pinned, it is also processed to form a vector $\mathbf{y} \in \mathbb{R}^d$. The goal is to check for any existing duplicates or near-duplicates to $\mathbf{y}$ in the database. Specifically, Pinterest would like to flag an image $\mathbf{q}$ as a near-duplicate to $\mathbf{y}$ if $\cos(\theta(\mathbf{q}, \mathbf{y})) \geq .98$. We want to find any near-duplicate with probability $\geq 99\%$.

Given this requirement, your job is to design a multi-table LSH scheme using SimHash to find candidate near-duplicates, which can then be checked directly against $\mathbf{y}$. To support this task, Pinterest has collected data on the empirical distribution of $\cos(\theta(\mathbf{q}, \mathbf{y}))$ for a typical new image $\mathbf{y}$. It roughly follows a bell-curve:



Pinterest wants to consider two possible computational targets for your LSH scheme, which will determine the speed of the de-duplication algorithm:

1. Ensure that no more than 1 million candidate near-duplicates are checked on average when a new image is pinned. Here "checked" means directly compared against the new image for high cosine similarity.

2. Ensure that no more than $200,000$ candidates are checked on average when a new image is pinned.

Based on the data above, describe how to set parameters for your LSH scheme to minimize the space (i.e., number of tables) used, while achieving each of the above goals. Justify your answers, and any assumptions you make. If you code anything up to help calculate your answer, please attach the code. As in lecture, you can assume that each hash table has $m = O(N)$ slots and this is large enough to ignore lower order terms depending on $1/m$.