NYU TANDON SCHOOL OF ENGINEERING

New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 9223D: Homework 2.
Due Friday, October 9th, 2020, 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: Johnson-Lindenstrauss Approximates Inner Products.

(10 pts) 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 $\geq 1 - \delta$.

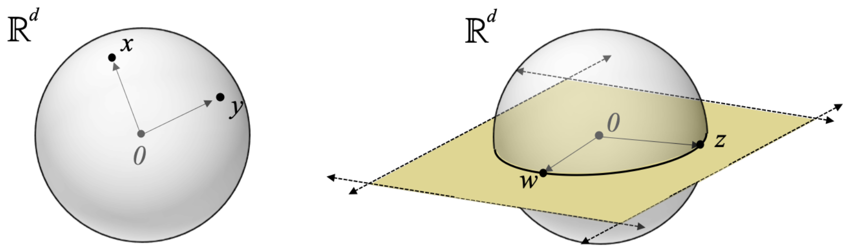
Problem 2: Geometry of high-dimensional vector pairs.

(10 pts)

1. Let x and y be random unit vectors in $\mathbb{R}^d$. Let $Z = \langle x, y\rangle$. Prove that $\text{Var}[Z] = \frac{1}{d}$. **Hint:** Since doing so will not change the distribution of $\langle x, y\rangle$, without loss of generality we can assume $x = e_1$ where e_1 is the first standard basis vector: $[1, 0, \ldots, 0]$.

2. Consider the following two random processes, which are illustrated with images below:

   - Pick two uniformly random unit vectors x, y from the sphere in d dimension.
   - Pick a uniformly random plane passing through the origin in d dimensions. Then pick two uniformly random unit vectors w, z from the 2D sphere lying on that plane.



   Are x and w identically distributed, or not? Are y and z identically distributed, or not? Are the pairs (x, y) and (w, z) identically distributed or not? Answer all three questions and argue your reasoning.

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?

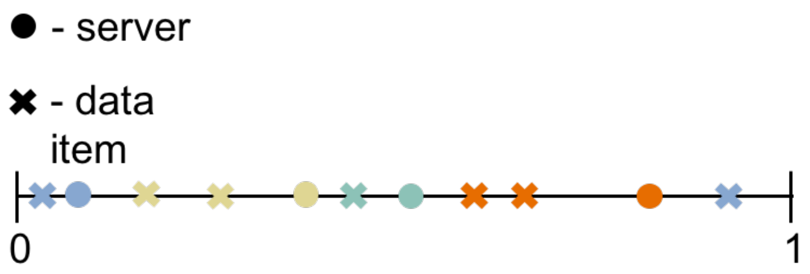


Figure 1: Each data item is stored on the server with matching color.

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 $O(\frac{m}{n}\log n)$ with probability $> 9/10$.

Problem 4a: 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 $> 9/10$).

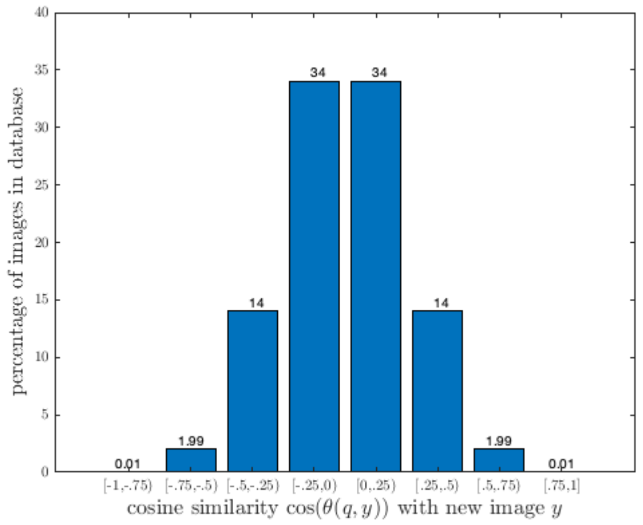
Problem 5: LSH in the Wild

Note: Based on topics that will be covered in the 9/30 lecture.

(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 $N =$ **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 class, you can assume that each hash table in your scheme is large: with $O(N)$ buckets.

(Relatively Easy) Bonus: Understanding Moment Bounds

(5 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. 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 "central" second moment $(X - \mathbb{E}[X])^2$.
3. What happens for higher moments? Try applying Markov's to X^q for $q = 3, 4, \ldots, 10$. Describe what you see. What value of q gives the tightest bound?
4. Exhibit a monotonic function g so that applying Markovs 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.