New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 9223I: Homework 2.
Due Thursday, October 17th, 2019, 11:59pm.

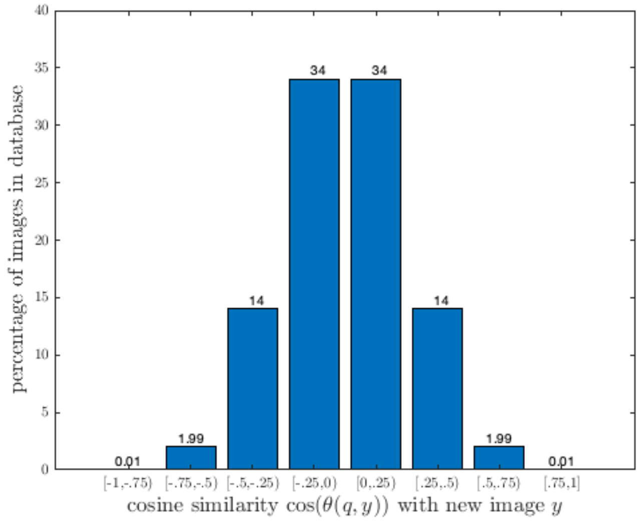
Collaboration is allowed on this problem set, but solutions must be written-up individually. Please list collaborators for each problem separately, or write "No Collaborators" if you worked alone.

Problem 1: LSH in the Wild

(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:



Use this data to determine:

1. How many tables does your LSH scheme require to ensure that no more than 1 million candidate near-duplicates are checked on average when a new image is pinned?
2. How many tables to ensure that no more than 200,000 candidates are checked?

Describe your LSH scheme at a high-level and provide numerical justification for your answers above. If any program is used 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.

Problem 2: Convex Sets.

(15 pts) A *convex set* $\mathcal{S}$ in $\mathbb{R}^d$ is a set of vectors with the follow property: if $\mathbf{x},\mathbf{y} \in \mathcal{S}$, then for all $\lambda \in [0,1]$,

$$[\lambda \mathbf{x} + (1-\lambda)\mathbf{y}] \in \mathcal{S}.$$

In other words, if $\mathbf{x}$ and $\mathbf{y}$ lie in $\mathcal{S}$, so does the entire line connecting them.

1. Prove that the following sets $\mathcal{S}$ are convex:
   (a) Let C be any scalar value. $\mathcal{S} = \{\mathbf{z} : \|\mathbf{z}\|_2 \leq C\}$
   (b) Let C be any scalar value and $\mathbf{a} \in \mathbb{R}^d$ be any vector. $\mathcal{S} = \{\mathbf{z} : \mathbf{z}^T\mathbf{a} \geq C\}$
   (c) Let $f : \mathbb{R}^d \to \mathbb{R}$ be any convex function. Let C be any scalar value. $\mathcal{S} = \{\mathbf{z} : f(\mathbf{z}) \leq C\}$.
2. Let $P_{\mathcal{S}} : \mathbb{R}^d \to \mathcal{S}$ be a "projection oracle" for $\mathcal{S}$. $P_{\mathcal{S}}$ is a function which maps any point in $\mathbb{R}^d$ to the nearest point in S:
$$P_{\mathcal{S}}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{S}} \|\mathbf{x} - \mathbf{y}\|_2.$$
   Show how to implement projection oracles for convex sets (a) and (b) described above.
3. Prove that, for any $\mathbf{v} \in \mathcal{S}$ and any $\mathbf{x} \in \mathbb{R}^d$,
$$\|\mathbf{v} - P_{\mathcal{S}}(\mathbf{x})\|_2 \leq \|\mathbf{v} - \mathbf{x}\|_2.$$

Problem 3: Gradient Descent with Constraints.

(5 pts) The version of gradient decent presented in class is easily modified to handle *constrained optimization problems*. In particular, suppose we wish to solve:
$$\min_{\mathbf{x}\in\mathcal{S}} f(x)$$
where $\mathcal{S} \subseteq \mathbb{R}^d$. Assume we have access to a projection oracle $P_{\mathcal{S}}$ for $\mathcal{S}$.

We can modify the update rule for gradient descent. Instead of setting $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} - \eta\nabla f(\mathbf{x}^{(i)})$, we set $\mathbf{x}^{(i+1)} = P_{\mathcal{S}}(\mathbf{x}^{(i)} - \eta\nabla f(\mathbf{x}^{(i)}))$. At the end of T steps, we return $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}^{(i)}} f(\mathbf{x}^{(i)})$.

1. Prove that, *when $\mathcal{S}$ is convex* this constrained version of gradient descent returns a point $\hat{\mathbf{x}}$ in $\mathcal{S}$ with $f(\hat{\mathbf{x}}) \leq \epsilon + \min_{\mathbf{x}\in\mathcal{S}} f(x)$ after $T = O\left(\frac{R^2G^2}{\epsilon^2}\right)$ iterations (exactly the same bound proven for unconstrained optimization).

Note that Online Gradient Descent can be modified in the same way, achieving the bound shown in class, even under a convex constraint.

Problem 4: Putting Online Gradient Descent to Work.

(5 pts) Read the **Portfolio Management Case Study** presented in Section 2.2 of the notes here. Suppose you wish to use Online Gradient Descent along with the Constantly Rebalanced Portfolio strategy. In particular, you will adjust over time which proportion of your money is kept in each of n assets. Let $\mathbf{x}^{(t)} \in [0,1]^n$ be a vector representing this allocation at time step t. Since you can only invest 100% of the money you have, and we assume you keep all your money in some asset, we always have that $\sum_{i=1}^n \mathbf{x}^{(t)} = 1$.

1. Derive an equation for the Online Gradient Descent update if your goal is to maximize total wealth increase over time. You can assume R, G, and your total number of time steps T are known ahead of time. Also assume you have access to a projection oracle P for the (convex) probability simplex: $\{\mathbf{z} : \text{for all } i,\ z_i \geq 0 \text{ and } \sum_{i=1}^n z_i = 1\}$ (a simple way of implementing one is discussed here).