Due date: March 04, 2025

Problem 1: Cosine Distance & LSH. In this problem we consider a family $\mathcal{F}$ of hash functions built upon the idea of cosine distance. Consider some high-dimensional space $\mathbb{R}^n$. Each hash function f in our collection is constructed from a randomly chosen vector $v_f \in \mathbb{R}^n$. Given two vectors x and y, we say $f(x) = f(y)$ if and only if the dot products $v_f^T x$ and $v_f^T y$ have the same sign.

(1) Describe one way to define f (i.e., given x, how do you define $f(x)$?).

(2) Describe a geometric interpretation of these hash functions. What does it mean if $f(x) = f(y)$? Explain why $\mathcal{F}$ can be viewed as a locality-sensitive family for the cosine distance on $\mathbb{R}^n$.

(3) Given $r \in (0, 1)$ and ϵ, find $p_1, p_2 \in (0, 1)$ such that $\mathcal{F}$ is $(r, (1+\epsilon)r, p_1, p_2)$-sensitive under cosine distance between v_f.

(4) Suppose now we want to amplify the sensitivity, i.e., aim for larger $p_1' < 1$ and smaller $p_2' > 0$. Describe how you would modify $\mathcal{F}$ or your previous solutions (or both) to achieve this. Try to be as detailed as possible, though the grading will prioritize right high-level ideas over the math.

Problem 2: ANN Query. Let S be a set of n points in $\mathbb{R}^2$, let q_i be the charge of point p_i in S, and let ϵ be a parameter. Assume that the spread of S is $n^{O(1)}$. Define

$$F(S) = \sum_i \sum_{j \neq i} \frac{q_i q_j}{\|p_i - p_j\|^2}.$$

Describe an $O(n\varepsilon^{-2} \log n)$ algorithm to compute an estimate $\tilde{F}(S)$ of $F(S)$ that has a multiplicative error of $\leqslant \epsilon$, i.e., $|\tilde{F}(S) - F(S)| \leq \varepsilon F(S)$. **Hint**: *Use the quad tree query process covered in lecture.*

Problem 3: Clustering Implementation. In this problem you will implement three k-means clustering algorithms on a few datasets. The guidelines are the same as in HW2 — you will submit a write-up as well as the source code. Please implement the following:

- **Lloyd**. Implement Lloyd's algorithm as described in L10. Start with k random centers, then iteratively alternate between assigning each point to their closest center and updating center to be the centroid of each cluster. Repeat until centers stabilize.
- **RandomizedGreedy**. Implement the randomized greedy algorithm as described in L9. Start by drawing k centers from the adaptive distance-to-closest-point distribution, then continue to perform at most ck steps of local search before terminating, where c is a constant specified by you.
- **CombinedAlgorithm**. Implement a combination of Lloyd's algorithm and randomized greedy, where one first obtains the k centers returned by RandomizedGreedy then feeds it into Lloyd's algorithm.

- Any helper function that you deem necessary, e.g. for parsing the dataset or for performing data analysis.

You are provided two datasets as `.csv` files, both of which can be found on Canvas. Each row in the csv file contains a list of d comma-separated numbers that represent the d-dimensional embedding of a data point.

- Pen-Based Handwritten Digits: sample size $n = 10992$, datapoint dimension $d = 16$, and number of clusters $k = 10$.
- Gaussian blobs, with $n = 10000$, $d = 5$, and *unknown* (well, to you at least) k.

Your write-up should contain the following. Because randomness is involved, consider repeating each experiment a few times before concluding. Parts (1)-(4) all use the Digits dataset.

(1) Assuming $k = 10$, plot the performance (clustering cost) of RandomizedGreedy change with respect to the number of local searches it performs.

(2) Assuming $k = 10$, compare Lloyd's algorithm against CombinedAlgorithm. Does seed selection (i.e. initial choices of clusters) matter?

(3) Report the performance of all three algorithms for every $k \in \{6, 8, 10, 12\}$. How does performance vary?

(4) Back to assuming $k = 10$, compare different termination conditions for Lloyd's algorithm, for example stopping after a fixed number of iterations when the cluster centers stabilize, or your own criteria. Which one(s) work best in practice, and why do you think this is the case?

(5) Test your algorithms on the Gaussian blobs dataset for various values of k. Report the clustering costs. What do you think is the right value of k? **Hint**: *the answer is not large*.