Homework 2

COL 761

- You need to do the homework in your already formed team of 3
- Due: **3rdth December 11:59PM.**
- **30%** penalty for each late day
- No multi-threading or parallelization is allowed
- Upload all code to GitHub and include a readme to clone your repo and other instructions like in HW1.

---

1. This problem is about the behaviour of a uniform distribution of points in high-dimensional spaces. Generate a dataset of 1 million random points in d-dimensional space (d varying as 1, 2, 4, 8, 16, 32, and 64). Assume that the points are uniformly distributed over [0,1] in each dimension and that the dimensions are independent. Choose 100 query points at random from the dataset. Examine the farthest and the nearest data point from each query. Compute the distances using L1, L2, and L∞. Plot the average ratio of farthest and the nearest distances versus d for the three distance measures. Make sure to not include the query point itself in the nearest data point computation. Explain the results. [20 points]
2. Dataset: https://csciitd-my.sharepoint.com/:u:/g/personal/sayanranu_iitd_ac_in/EevtYg7xNK9OoG7zM-RVXOcBTCE1foRa5j-tTprfabYMzQ?e=W9qLVR
   This question is about the behaviour of hierarchical space partitioning technique R*-tree vs LSH on high dimensional data on very large datasets. Take the above image data, and perform *k*-NN (L2) on 100 random points. Plot the number of *inspections* and running time against:
   a. Dimensions: Reduce the number of dimensions to 2,4, 8, and 10 using PCA at k=1. To reduce dimension, if computing the co-variance matrix is too expensive on the entire dataset, you can sample a subset, compute eigen-vectors and then project all remaining points on those eigenvectors (10 points)
   b. Dataset size: on a subset of 1000 points, 10000 points, 100000 points, and the entire dataset at k=1 *at dimensions=[2,8]*. (10 points)
   c. Against *k=[1,5,50,100,500] at dimension=[2,8] in a subset of 100,000 points. [10 points].*
   d. Explain the results of a-c (20 points)

   For R*-tree (a variant of R-tree), the number of inspections is the number of points (or nodes) that were inspected. For LSH, it is the number of points that collided in at least one table (across all bits).

   Parameters: Play around with few samples to identify the optimal ones. However, make sure, that the query points and the points used to set the parameters are disjoint (like in classification where you don't use the test set to train the model). You can set r in LSH based on what is the "typical" distance of a 1-NN. Whatever procedure you use, make sure to document that. For R*-tree, use the in-memory version where the entire tree including the

data points are stored on memory. Same for LSH, where all the hash tables are stored in memory. The R*-tree should be built by *bulk-loading* the dataset.

Libraries:

- R*-tree: several libraries are available online. I would strongly urge you to use a C++ or Java based library since they are much faster than implementation in other languages. The library you use should support dimensions higher than 2, and in-memory storage. One option is provided below:
  - https://libspatialindex.org/en/latest/
- LSH: Once again several libraries are available. I suggest using https://github.com/FALCONN-LIB/FALCONN/wiki/LSH-Primer