

Homework 3, 290D, Fall 07
Due November 30, 5 pm
(2-person group homework)

1. (15 points) The aim of this problem is to understand the effect of increasing the number of dimensions on the volume of a hypersphere with constant radius. Compute the volume of a hypersphere of radius 0.5 for number of dimensions = [1,2,4,8,16,32,64]. Assuming a uniform distribution of points over the unit cube, how large the data size need to be to find one point within this hypersphere? Report the table of volumes and data sizes.
2. (15 points) The aim of the problem is to understand the distribution of distance between random points as the number of dimensions is increased. First generate 1000 random points from an i.i.d. uniform distribution with the number of dimensions (d) varying over [1,2,4,8,16,32,64]. For each value of d, compute and plot the distribution of all possible L2 distances. Compute the mean and variance of the distribution. Repeat for L1 distance. Report the means and variances for the two distance measures for the varying number of dimensions. What do you infer from the distributions? What would you expect to see if the distribution is Gaussian instead of uniform?
3. (15 points) Suppose we have a (disk) data page that is defined by a hypercube of side length 0.5 in d dimensions varying over [1,2,4,8,16,32,64]. Assume that 10^6 points have been packed within the unit cube. Consider two kinds of queries:
 - a. A hypercube range query that returns 10 points.
 - b. A NN query that returns 1 point.Compute the probability that the data page will be accessed for the two kinds of queries for the varying number of dimensions.
4. (55 points) This problem is a continuation of the image dataset problem from the previous homework. You will be working with the image dataset located at CSIL in the “/cs/class/cs290d/sandbox/homework2/cortina_images” directory. You will be working the SCD descriptor reduced to 8 dimensions using PCA. Compute an index on the points using
 - a. VA-file. 8 bits per dimension.
 - b. R-tree. Simulate this in main memory assuming a page size of 4KB. Choose your own strategy for defining the MBRs. You can compute this statically.

Implement an algorithm for top-k query. With a set of query points that will be provided to you, compute the top-10 neighbors. Report the objects returned and the time taken to compute. Assume that a sequential page access takes 0.01 ms and a random page access takes 10 ms. Compare the times with sequential scan.