Mahardiansyah Kartika
mkartika@andrew.cmu.edu

Kd-Tree Exercise

The purpose of this exercise is to create Kd-Tree from input file such as .csv and performing efficient nearest neighbor search over the tree. There are several steps in this process.

1. Kd-Tree Construction

Tree is built by N-dimensional point cloud from input file. If there is only one point, create a leaf with that point. Otherwise, divide the point based on splitting axis (largest range dimension) and splitting point (median value of points from splitting axis). They will do it recursively until the node only has one point or until the node reach certain depth. In this case, I set 30 as the maximum depth for the tree. The maximum depth will anticipate the size of the tree if the point cloud contains millions points. The tree depth definition can be found in `tree.h` file in line 15. `#define MAX_TREE_DEPTH 30`

By default, median will be chosen as splitting point. However, there are two ways to set splitting point. It can use median or mean value. It can be switch by change `tree.h` file in line 54.
`split_type = SplitType::MEDIAN` or `split_type = SplitType::MEAN`

Moreover, for N-dimensional point value, it supports `float` and `double` format. By default, it uses `float` as the type format. Because `float` consumes less memory compared to `double`. Although, `double` has double-precision, however for current `sample_data` and `query_data`, single-precision is sufficient to perform searching-nearest-neighbor. Take a look at `math.h` in line 20, comment the line if want to switch to `double` format. `#define REAL_FLOAT 1`. Constructing Kd-Tree has O(dn log n) time complexity and O(dn) space complexity.

2. Search Nearest Neighbor

Searching nearest neighbor is done by greedily descent through the tree and also backtracking through the ignorance branches of the tree. By storing the minimum distance to best found leaf nodes, we can compare with the distance to the hyperplanes. If the distance to hyperplanes is bigger than the distance to the best found node, then we can ignore the node on the other side of the hyperplanes entirely.

Time complexity for this search is O(log n) average time per search in a reasonable model. A simple heuristic choosing split axis that has largest variation and choosing median value as splitting point makes almost balanced tree and we can see balanced tree as a reasonable model.

3. Future Improvements

For small sample data, this algorithm works well. But if we add a lot of points to sample data, the algorithm will get slower because the sorting algorithm to find the median value. It will be better to do the preprocessing to sort the points in each dimension. It requires linear space complexity but it will make the algorithm run faster. And also, using squared-distance rather than distance to calculate the length of the two points will make the algorithm run faster.