Kdtree: Design and Implementation

Russ Hamilton
May 26, 2016

Overview

- Provides classes and functions for
  - Reading k-dimensional points from a CSV from a stream
  - Generating a kd-tree from the set of points
    - kd-tree is optimized for nearest neighbor search
  - Serializing the kd-tree to a stream
  - Deserializing the kd-tree from a stream
  - Efficiently find the nearest neighbor in the tree to an arbitrary point

Tree Representation

- Point coordinates
  - Kept in flat array
  - Calculations refer to points indirectly by using their index in the array
  - Reduces the memory footprint and amount of copying required to build the tree
- Tree structure
  - Array of nodes (with the same length as the array of points)
  - Each node keeps track of
    - The axis to split over
    - The index of this node
    - The index of its left and right children
  - Keeping indices instead of references make serialization easy

Building the Tree

- Recursive algorithm
  - Takes the list of points
  - Finds the axis with largest difference
  - Sorts points (indirectly) along this axis
  - Splits the list at the median point
  - Creates a node
    - Left child is the subtree generated by the list of points before the median
    - Right child is the subtree generated by the list of points after the median

Building the Tree

- Complexity - O(kn log n + n $\log^2$ n)
  - Finding the axis with the largest difference O(kn)
    - Assume std::minmax_element is O(n)
  - Sort points along a single axis O(n log n)
  - Recursive algorithm runs O(log n) times
- Potential optimization (not implemented)
  - Pre-sort the list of points across all dimensions
    - Makes finding axis with largest difference O(k)
    - No sort required
    - Overall complexity O(kn log n)

Finding the Nearest Neighbor

- Recursive algorithm to find nearest neighbor within bound
  - If current node is closest so far, update the closest
  - Recurse with subtree that would contain the target (if any), update the closest
  - Recurse with other subtrees if the region could contain a point closer than the current closest, update the closest

Finding the Nearest Neighbor

- Complexity – O(k^2 log n)
  - Point distance and region calculations – O(k)
  - Number of subtrees containing the target – O(log n)
  - Number of subtrees not containing the target that contain points closer to the target than the subtree containing the target
    - Worst case O(n)
    - Typical case O(k log n)

Summary

- Provides moderately efficient tree generation
- Provides efficient exact nearest neighbor search