KD-trees for nearest neighbor search

See also the animations in the ppanim directory!!

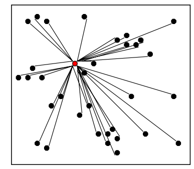
The Auton Lab

Carnegie Mellon University

www.autonlab.org

Nearest Neighbor - Naïve Approach

- Given a query point X.
- Scan through each point Y:
  - Calculate the distance d(X,Y)
  - If d(X,Y) < best_seen then Y is the new nearest neighbor.
- Takes O(N) time for each query!

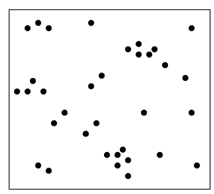


33 Distance Computations

Speeding Up Nearest Neighbor

- We can speed up the search for the nearest neighbor:
  - Examine nearby points first.
  - Ignore any points that are further then the nearest point found so far.
- Do this using a KD-tree:
  - Tree based data structure
  - Recursively partitions points into axis aligned boxes.

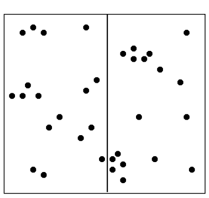
KD-Tree Construction



Pt	X	Y
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
…	…	…

We start with a list of n-dimensional points.

KD-Tree Construction



X>.5

NO

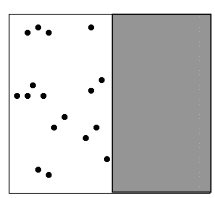
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

YES

Pt	X	Y
2	1.00	4.31
…	…	…

We can split the points into 2 groups by choosing a dimension X and value V and separating the points into X > V and X <= V.

KD-Tree Construction



X>.5

NO

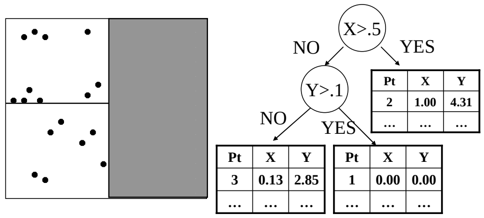
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

YES

Pt	X	Y
2	1.00	4.31
…	…	…

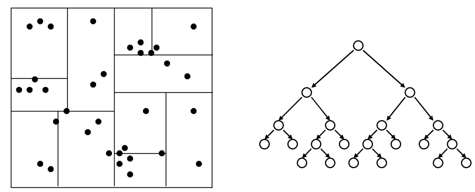
We can then consider each group separately and possibly split again (along same/different dimension).

KD-Tree Construction



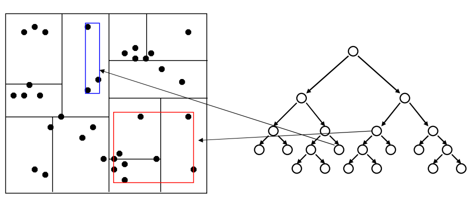
We can then consider each group separately and possibly split again (along same/different dimension).

KD-Tree Construction



We can keep splitting the points in each set to create a tree structure. Each node with no children (leaf node) contains a list of points.

KD-Tree Construction



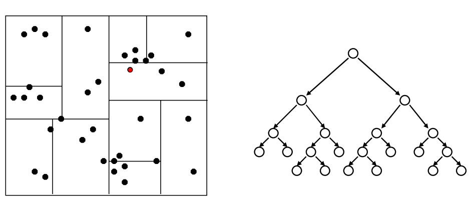
We will keep around one additional piece of information at each node. The (tight) bounds of the points at or below this node.

KD-Tree Construction

Use heuristics to make splitting decisions:

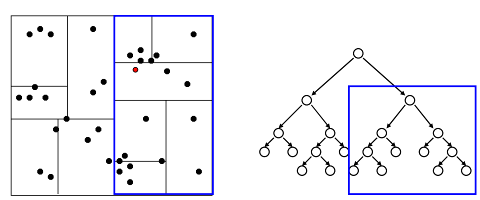
- Which dimension do we split along? Widest
- Which value do we split at? Median of value of that split dimension for the points.
- When do we stop? When there are fewer then k points left OR the box has hit some minimum width.

Nearest Neighbor with KD Trees



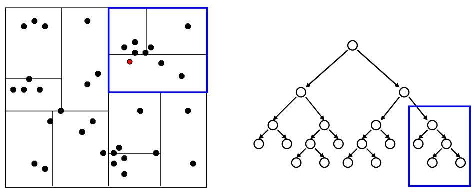
We traverse the tree looking for the nearest neighbor of the query point.

Nearest Neighbor with KD Trees



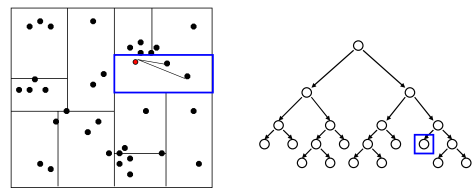
Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Nearest Neighbor with KD Trees



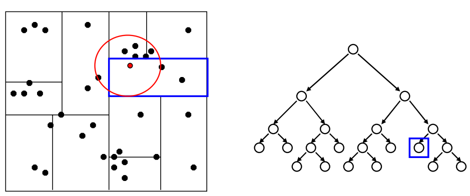
Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Nearest Neighbor with KD Trees



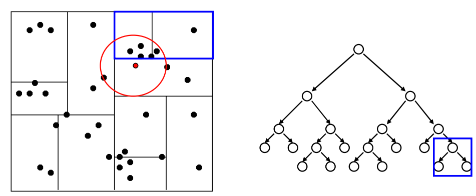
When we reach a leaf node: compute the distance to each point in the node.

Nearest Neighbor with KD Trees



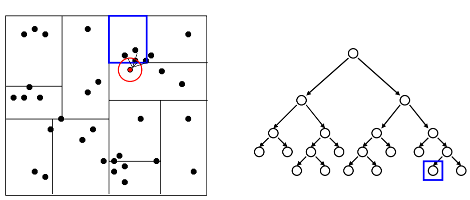
When we reach a leaf node: compute the distance to each point in the node.

Nearest Neighbor with KD Trees



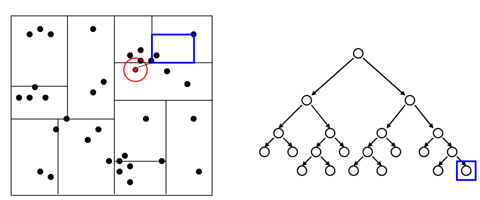
Then we can backtrack and try the other branch at each node visited.

Nearest Neighbor with KD Trees



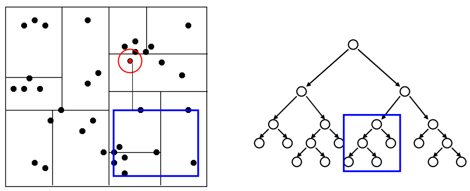
Each time a new closest node is found, we can update the distance bounds.

Nearest Neighbor with KD Trees



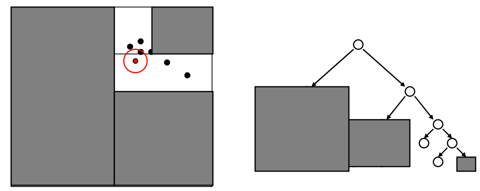
Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.