“Geometric” data structures:

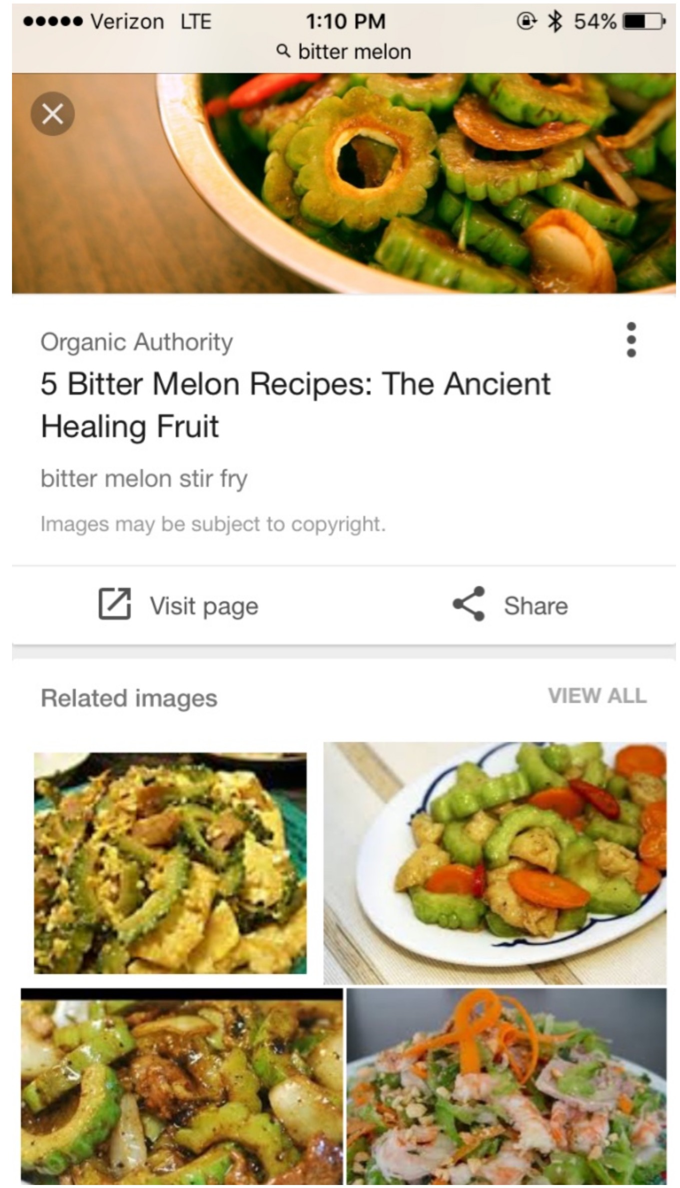
Machine Learning for Big Data
CSE547/STAT548, University of Washington
Sham Kakade

Announcements:

- HW3 posted

- Today:
  - Review: LSH for Euclidean distance
  - Other ideas: KD-trees, ball trees, cover trees

Image Search…



LSH for Euclidean distance

- The family of hash functions:
- Recall R, cR, P1, P2
- Pre-processing time:
- Query time:

What other guarantees might we hope for?

- Recall sorting:

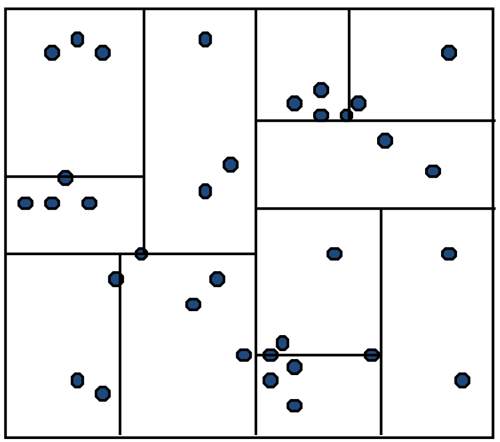
- LSH:
- Voronoi:

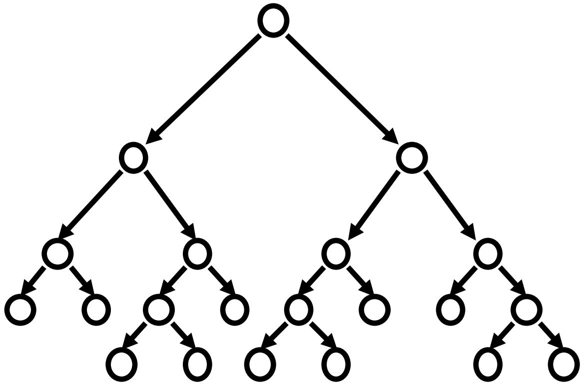
- How about other ”geometric” data structures?

- What is the ‘key’ inequality to exploit?

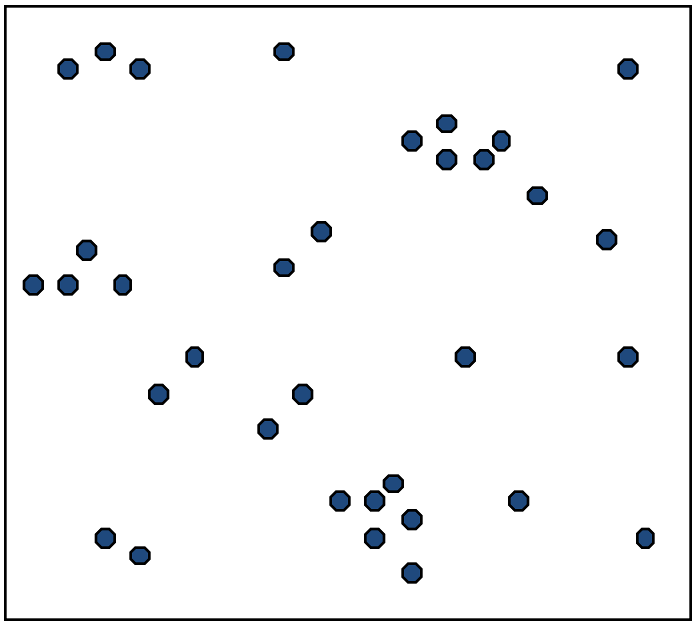
KD-Trees

- Smarter approach: ***kd-trees***
  - Structured organization of documents
    - Recursively partitions points into axis aligned boxes.
  - Enables more efficient pruning of search space
    - Examine nearby points first.
    - Ignore any points that are further than the nearest point found so far.
- ***kd-trees*** work "well" in "low-medium" dimensions
  - We'll get back to this…





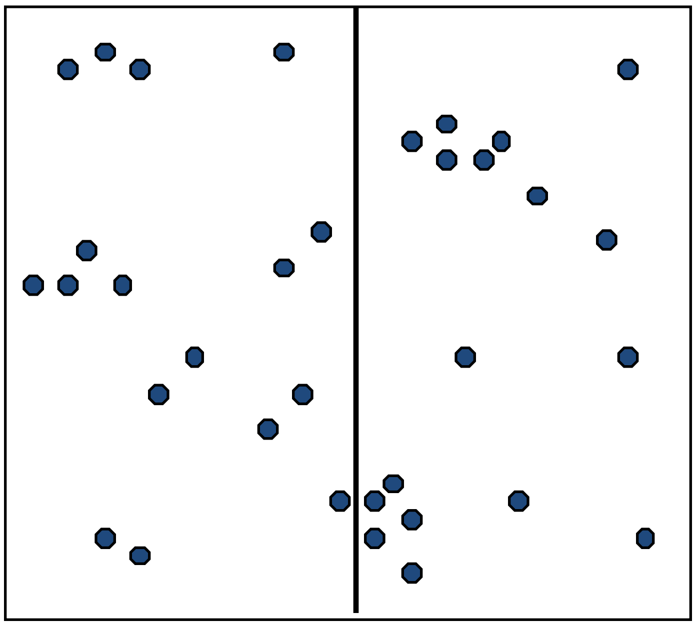
KD-Tree Construction

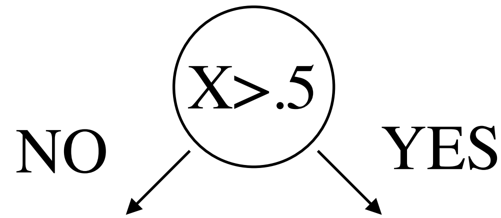


Pt	X	Y
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
…	…	…

- Start with a list of *d*-dimensional points.

KD-Tree Construction



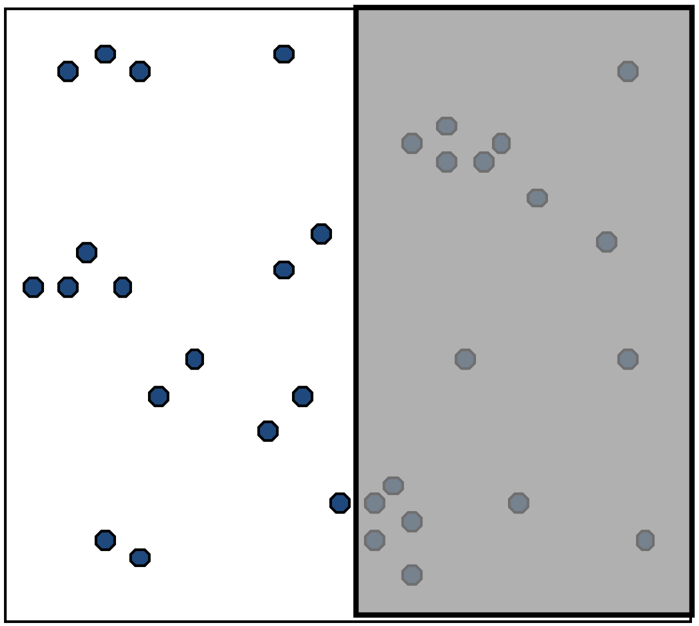


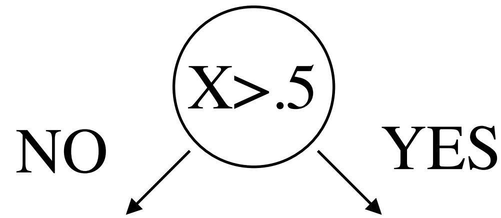
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	1.00	4.31
…	…	…

- Split the points into 2 groups by:
  - Choosing dimension d_j and value V (methods to be discussed…)
  - Separating the points into $x^i_{d_j}$> V and $x^i_{d_j}$<= V.

KD-Tree Construction



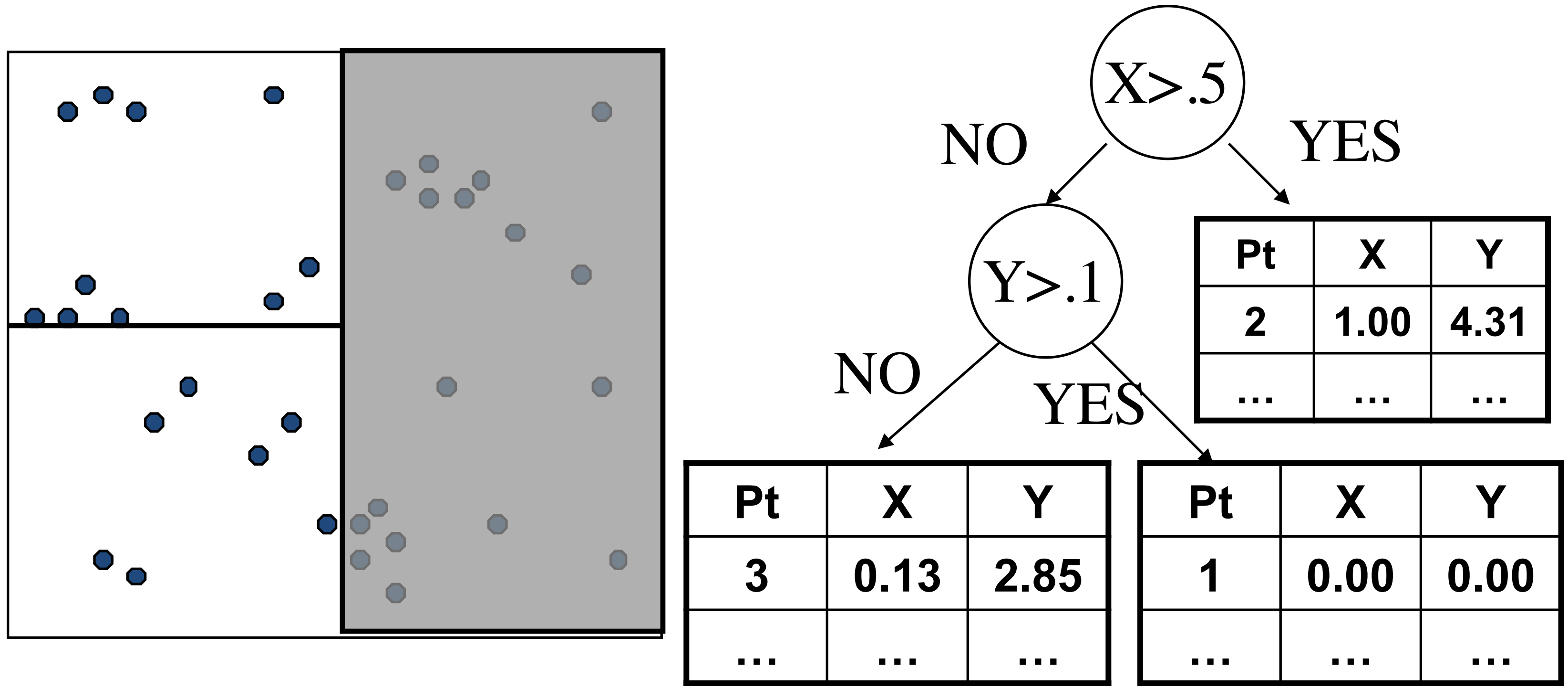


Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	1.00	4.31
…	…	…

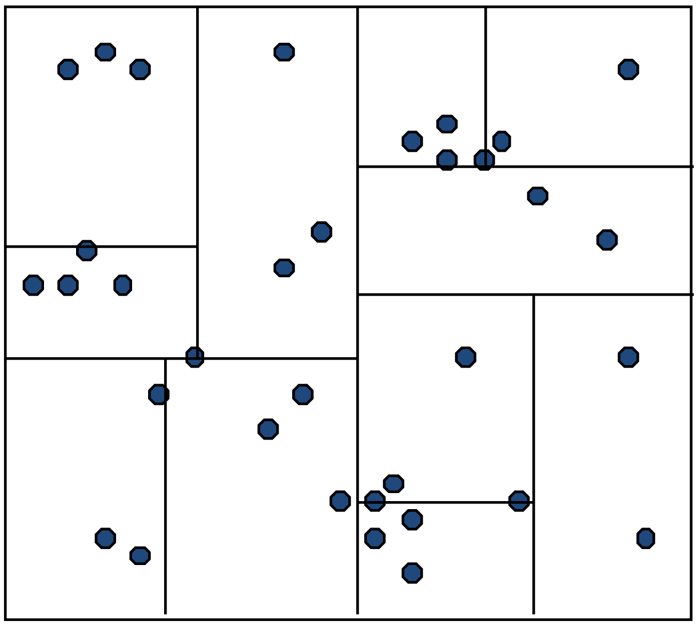
- Consider each group separately and possibly split again (along same/different dimension).
  - Stopping criterion to be discussed…

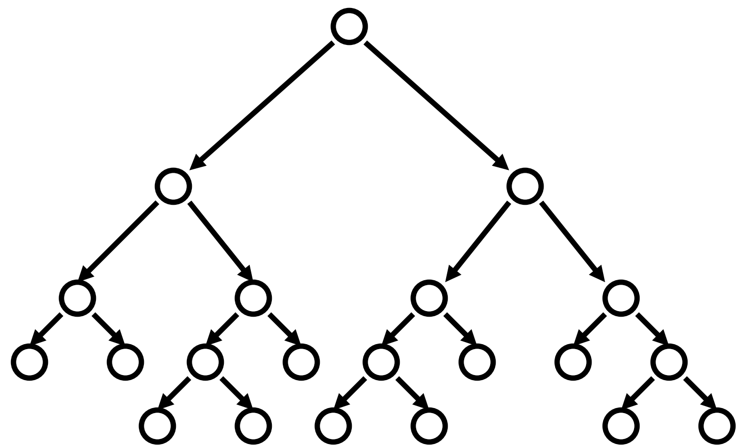
KD-Tree Construction



- Consider each group separately and possibly split again (along same/different dimension).
  - Stopping criterion to be discussed…

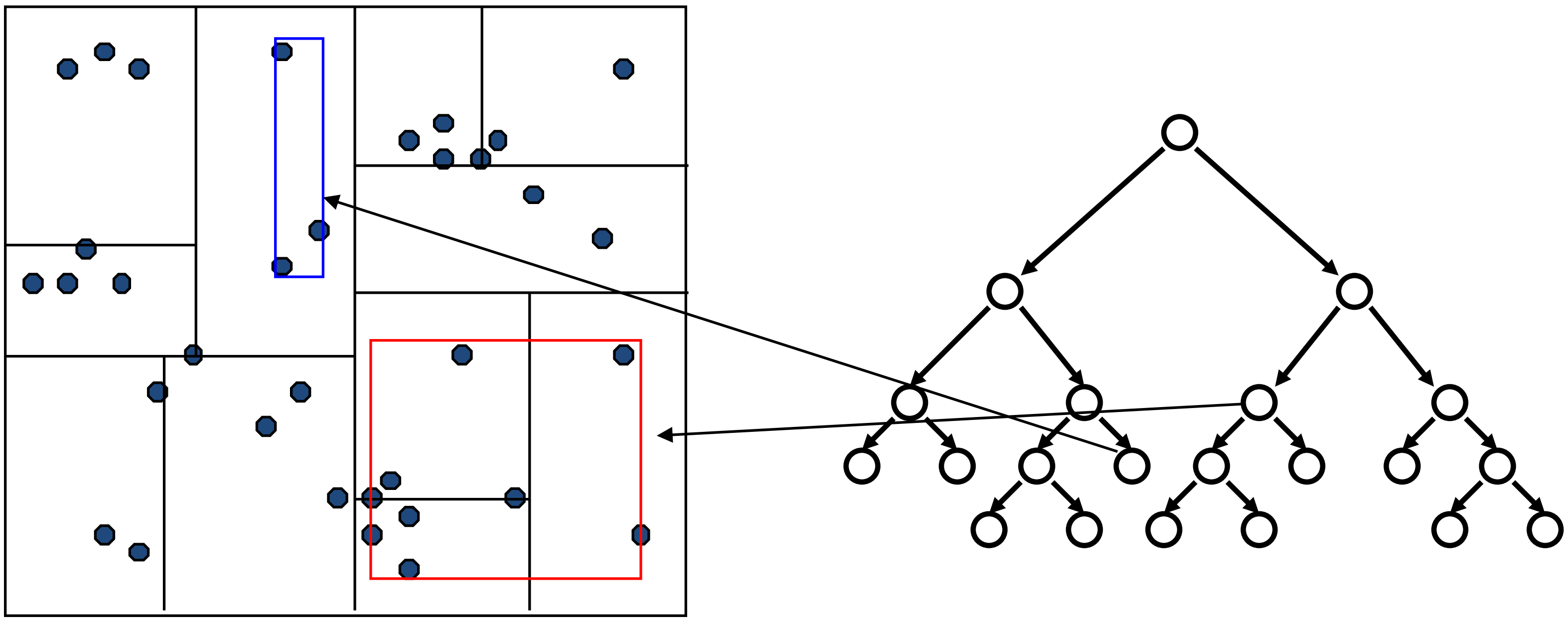
KD-Tree Construction





- Continue splitting points in each set
  - creates a binary tree structure
- Each leaf node contains a list of points

KD-Tree Construction



- Keep one additional piece of information at each node:
  - The (tight) bounds of the points at or below this node.

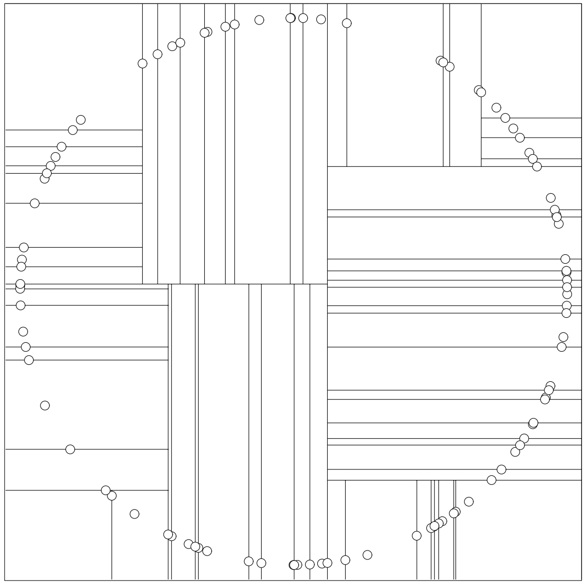
KD-Tree Construction

- Use heuristics to make splitting decisions:
- Which dimension do we split along?

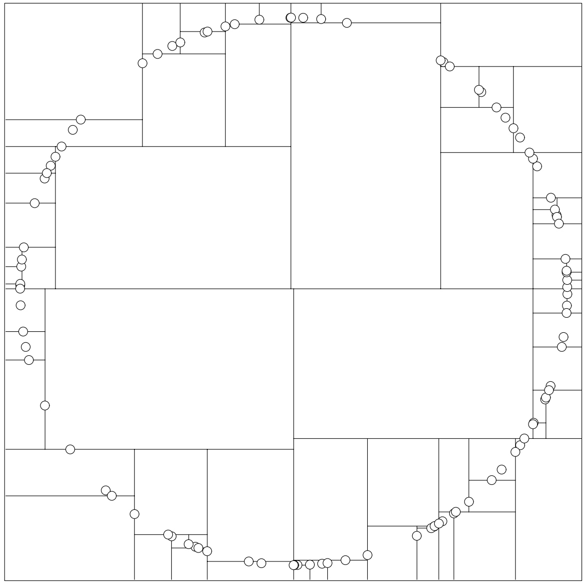
- Which value do we split at?

- When do we stop?

Many heuristics…

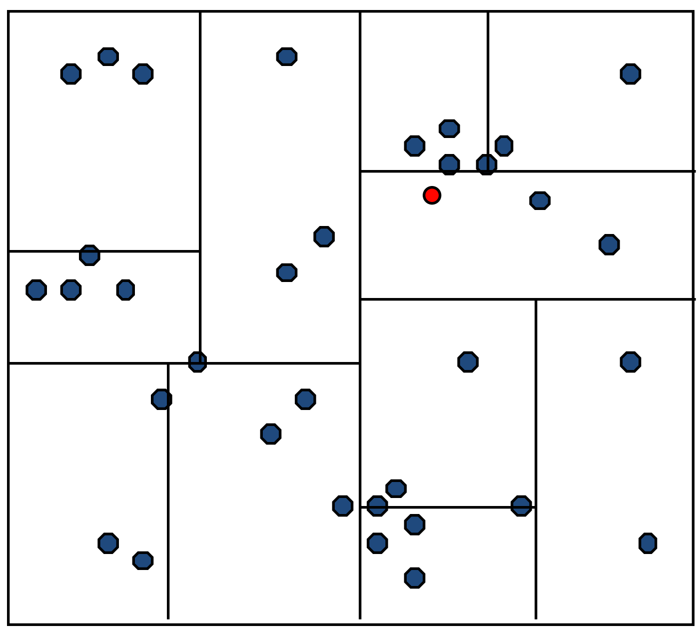


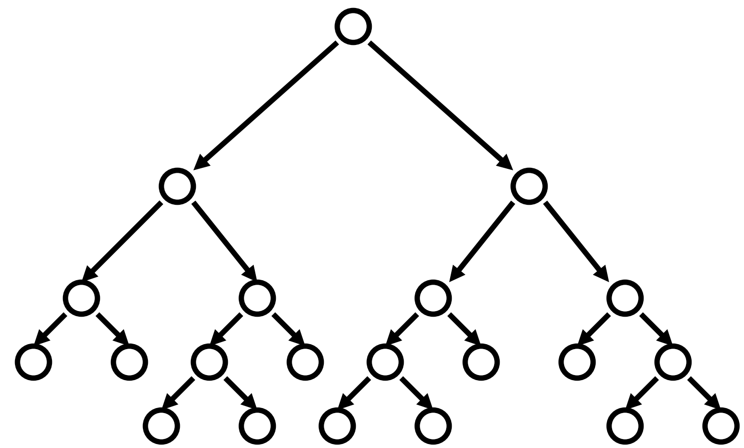
median heuristic



center-of-range heuristic

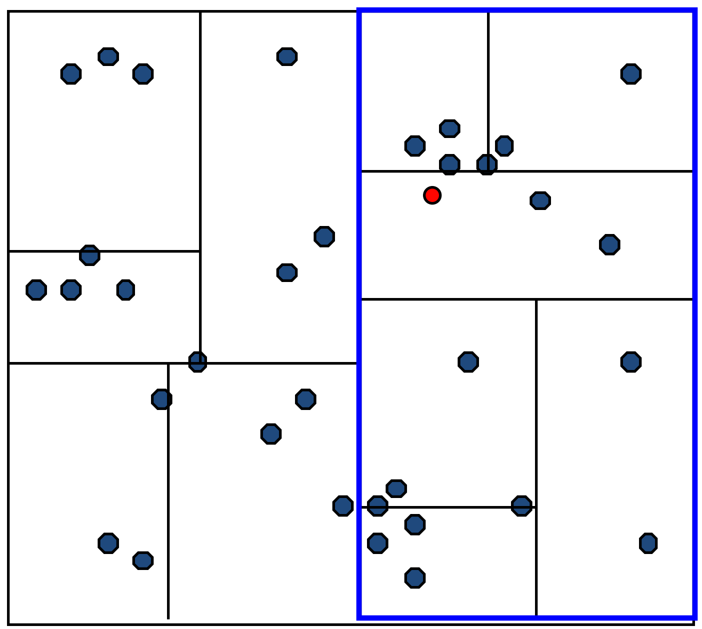
Nearest Neighbor with KD Trees

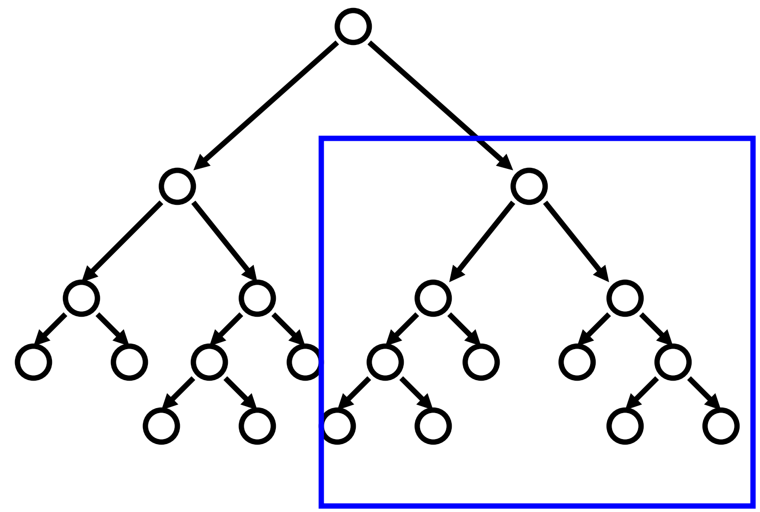




- Traverse the tree looking for the nearest neighbor of the query point.

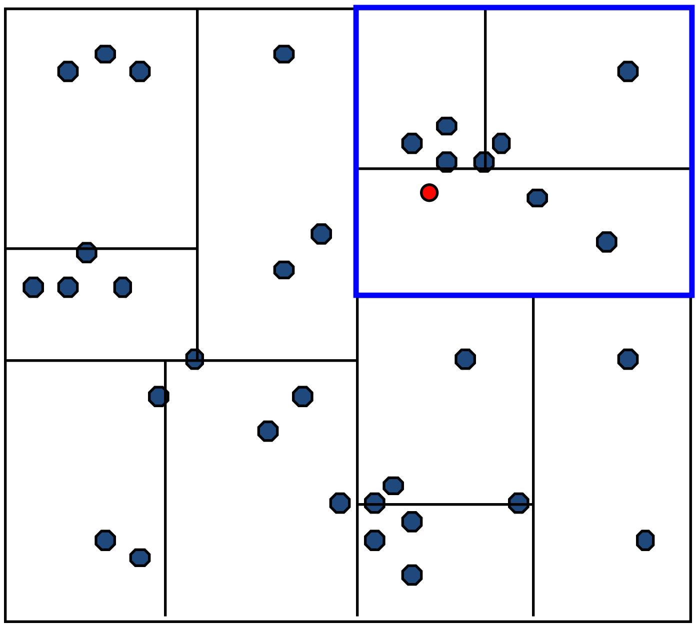
Nearest Neighbor with KD Trees

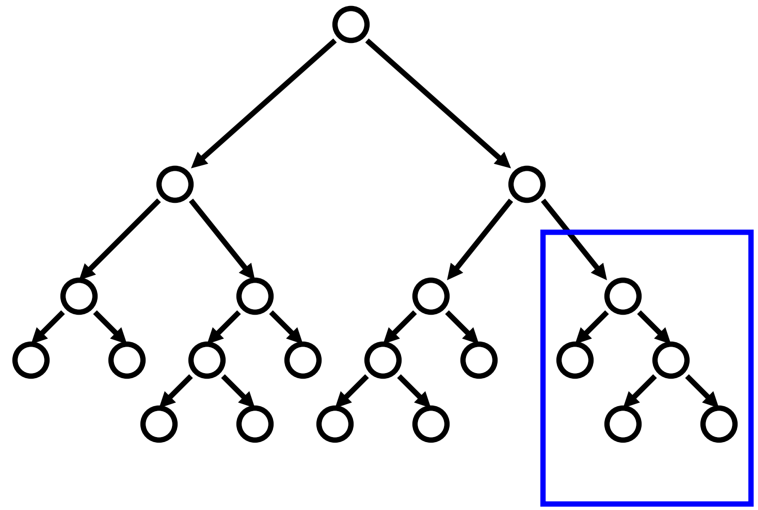




- <u>Examine nearby points first</u>:
  - Explore branch of tree closest to the query point first.

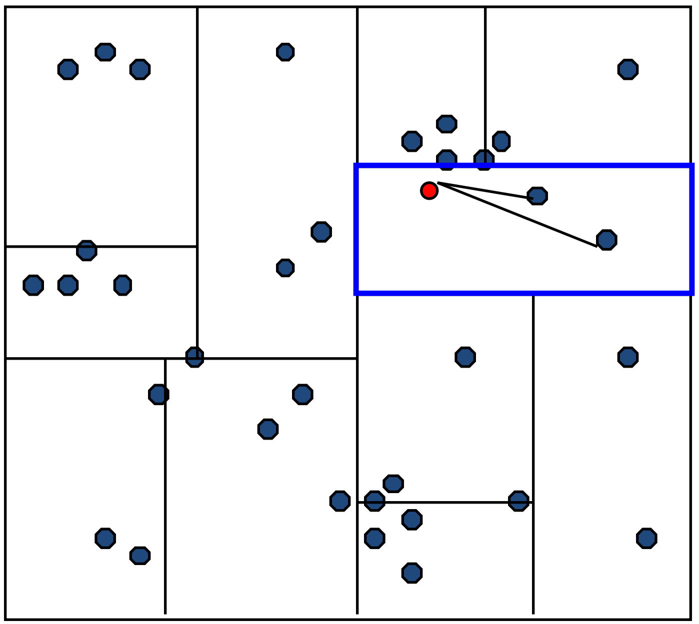
Nearest Neighbor with KD Trees

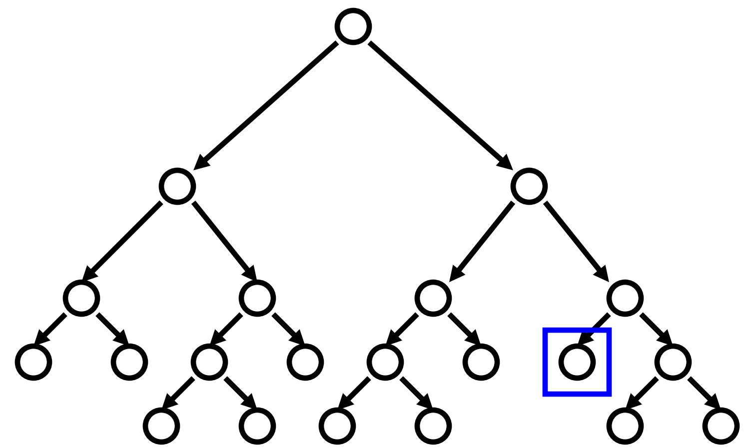




- Examine nearby points first:
  - Explore branch of tree closest to the query point first.

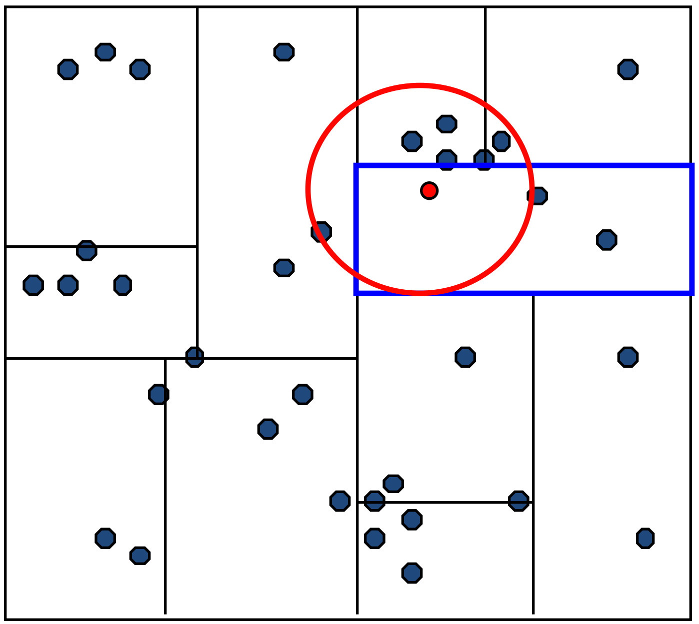
Nearest Neighbor with KD Trees

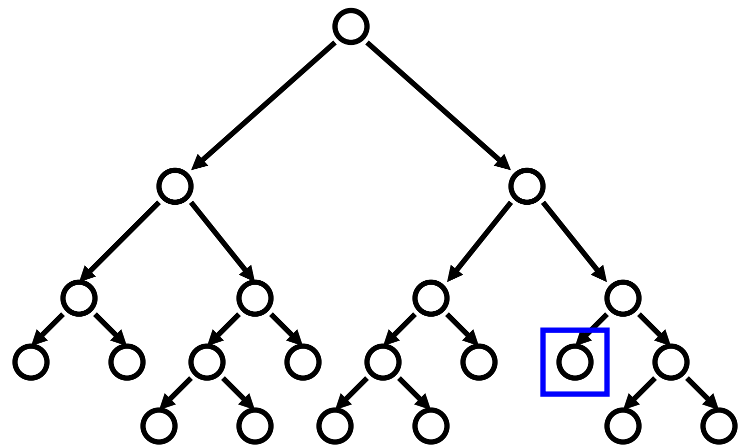




- When we reach a leaf node:
  - Compute the distance to each point in the node.

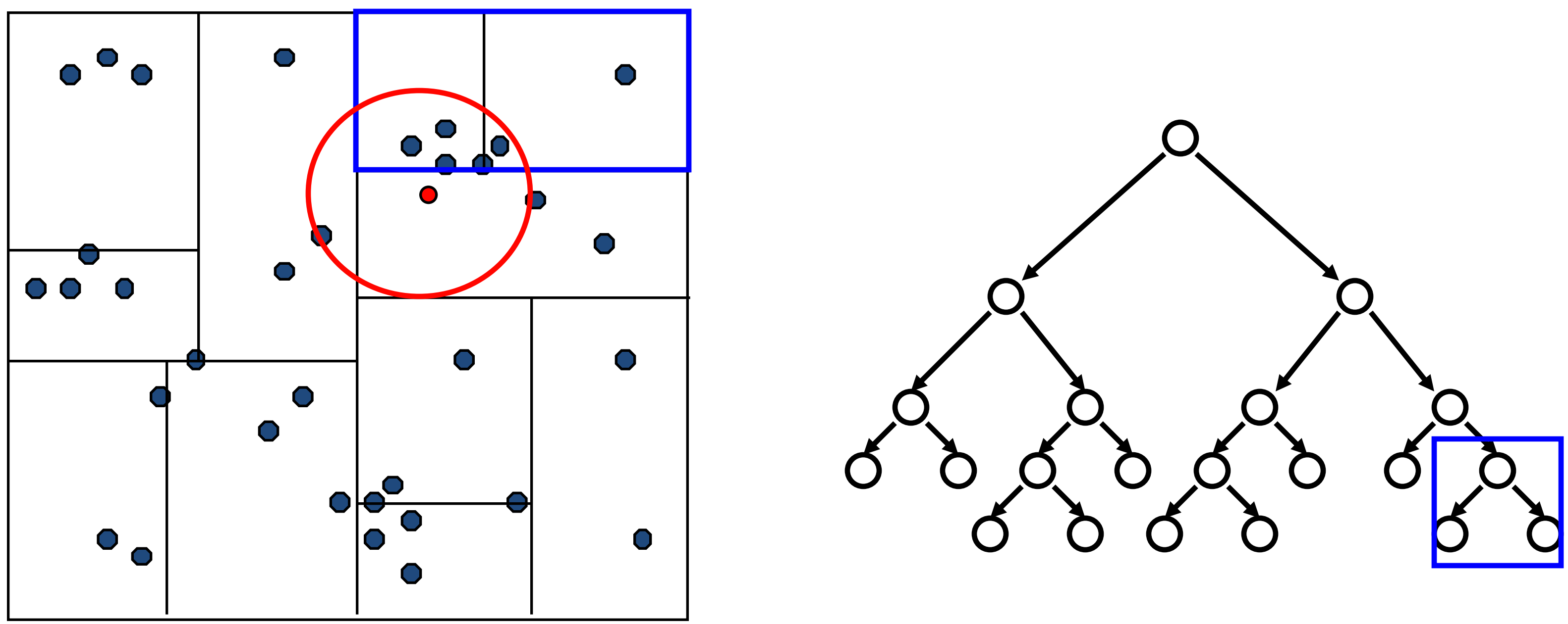
Nearest Neighbor with KD Trees





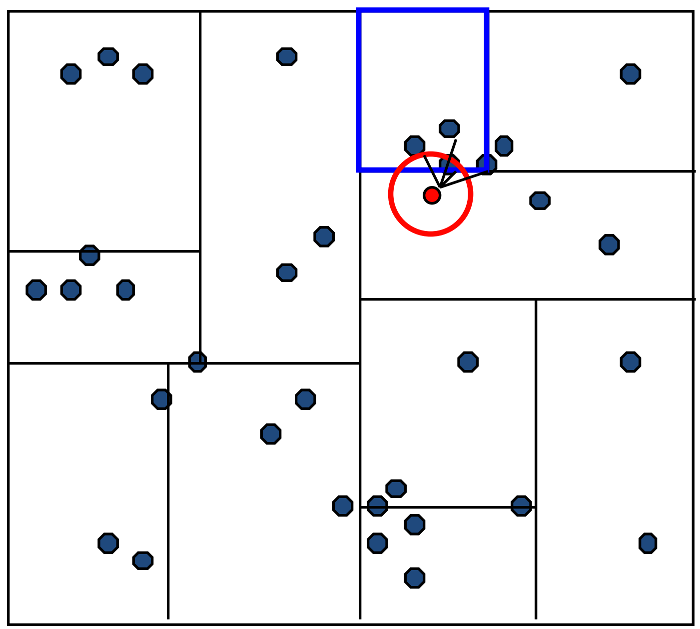
- When we reach a leaf node:
  - Compute the distance to each point in the node.

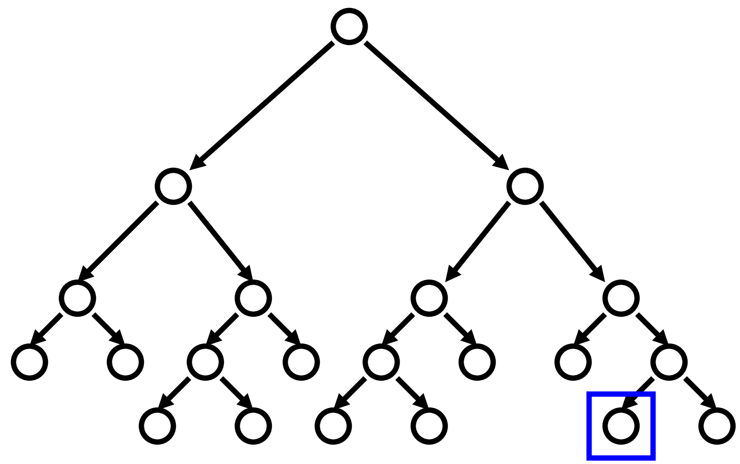
Nearest Neighbor with KD Trees



- Then backtrack and try the other branch at each node visited

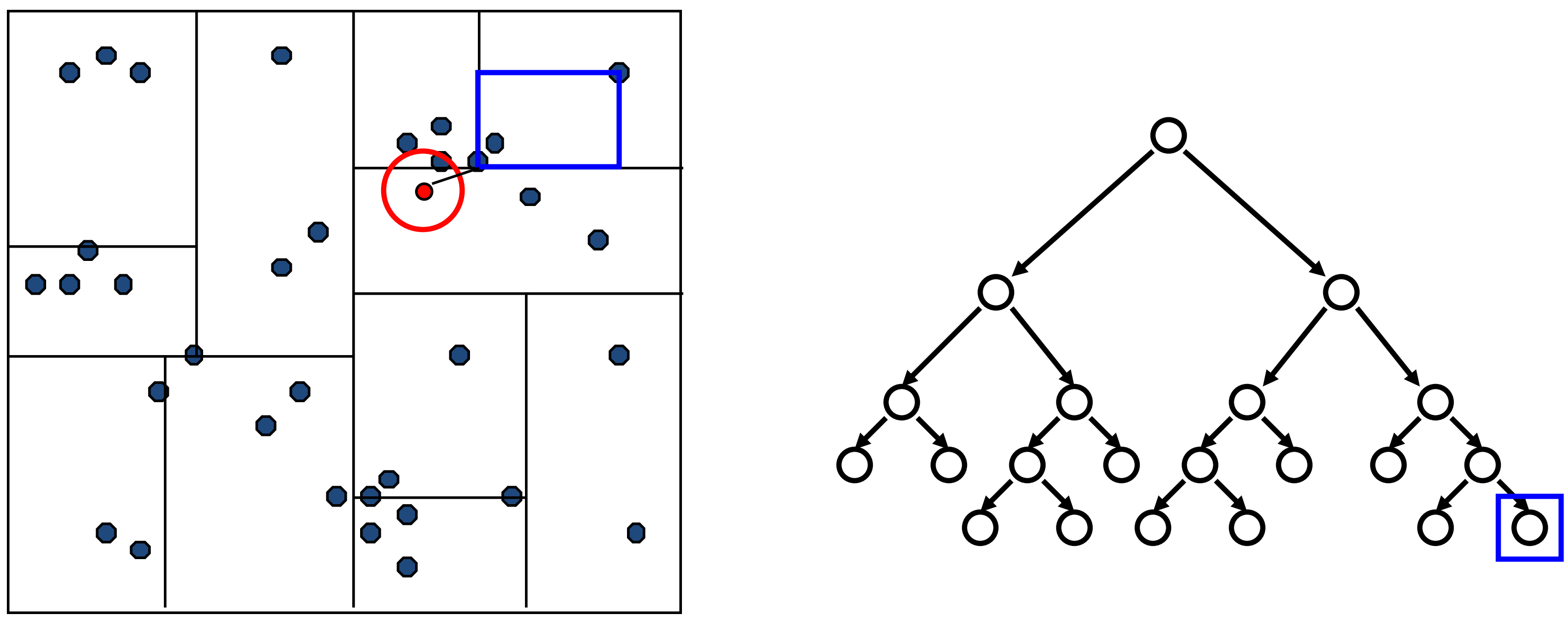
Nearest Neighbor with KD Trees





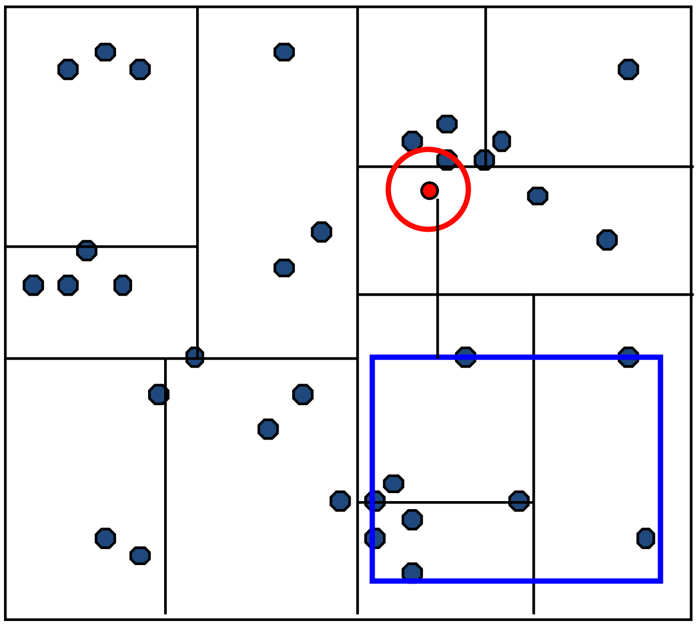
- Each time a new closest node is found, update the distance bound

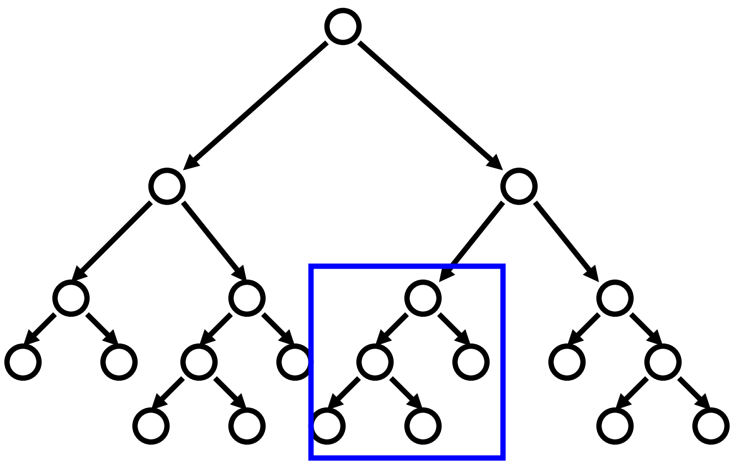
Nearest Neighbor with KD Trees



- Using the distance bound and bounding box of each node:
  - Prune parts of the tree that could NOT include the nearest neighbor

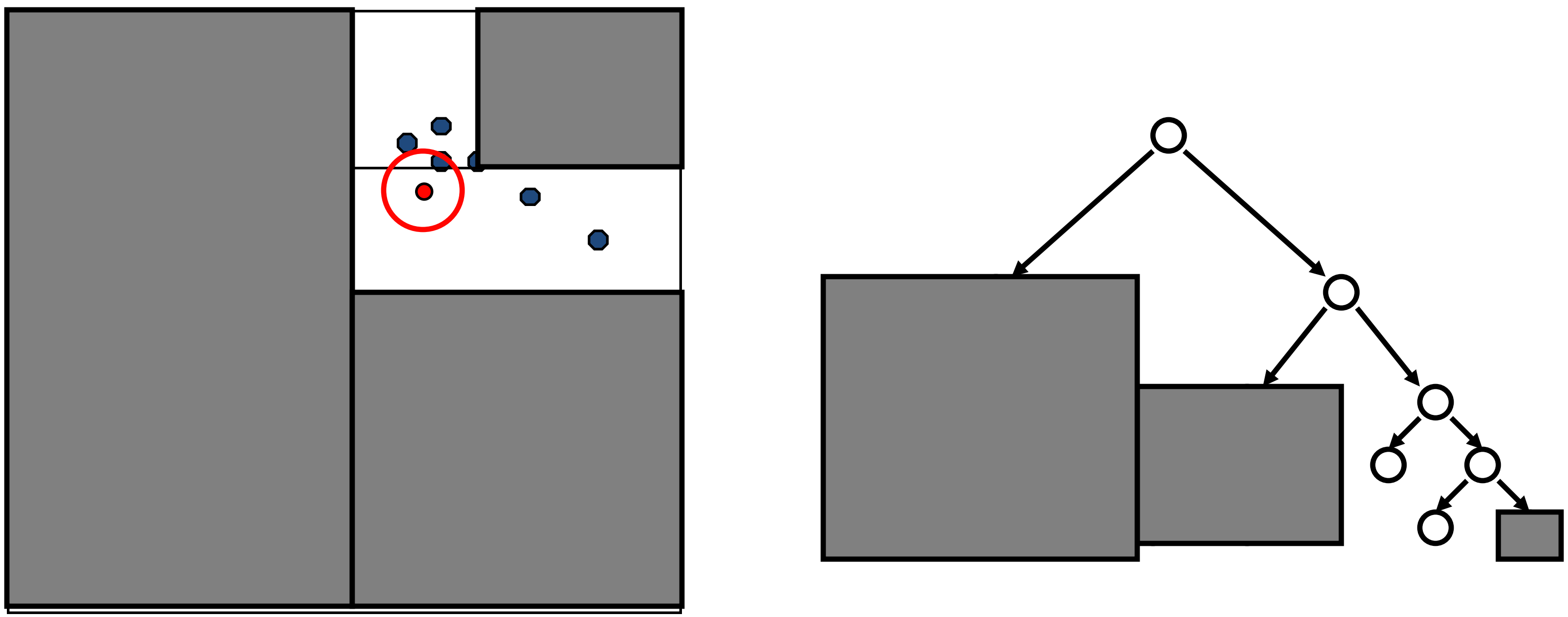
Nearest Neighbor with KD Trees





- Using the distance bound and bounding box of each node:
  - Prune parts of the tree that could NOT include the nearest neighbor

Nearest Neighbor with KD Trees

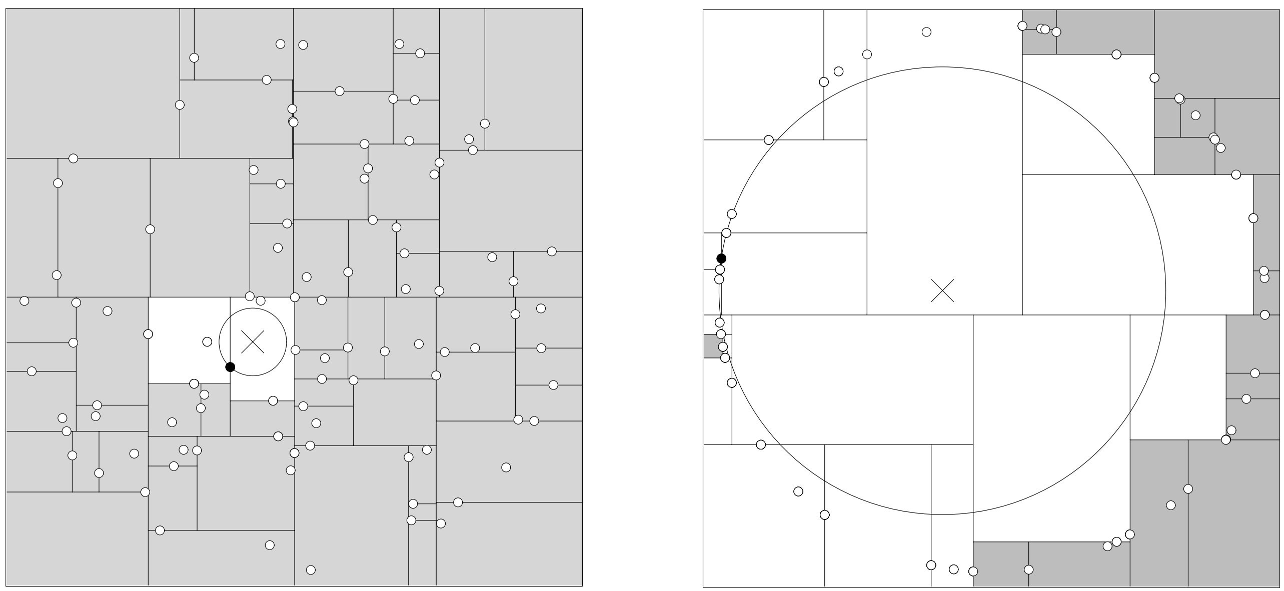


- Using the distance bound and bounding box of each node:
  - Prune parts of the tree that could NOT include the nearest neighbor

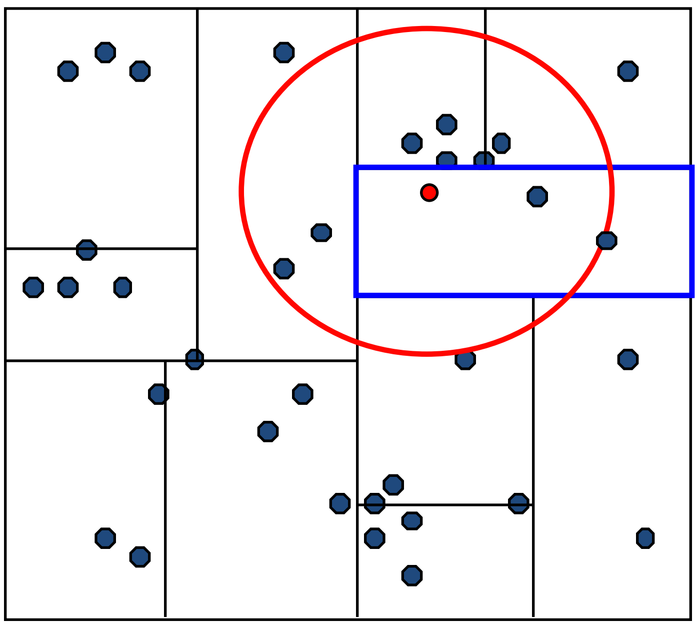
Complexity

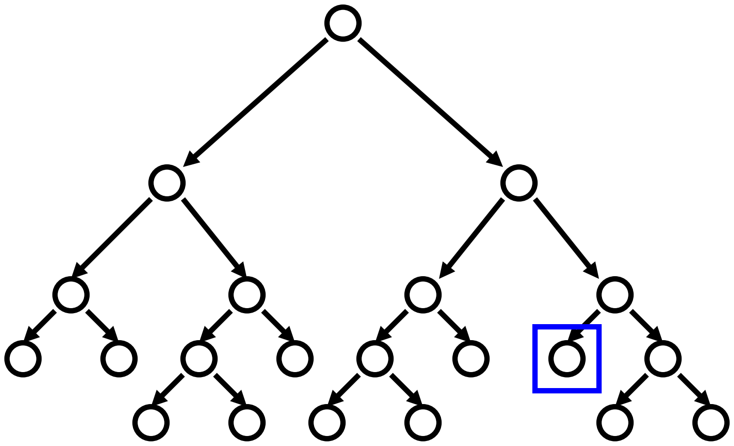
- For (nearly) balanced, binary trees...
- Construction
  - Size:
  - Depth:
  - Median + send points left right:
  - Construction time:
- 1-NN query
  - Traverse down tree to starting point:
  - Maximum backtrack and traverse:
  - Complexity range:
- Under some assumptions on distribution of points, we get O(log*N*) but exponential in *d* (see citations in reading)

Complexity



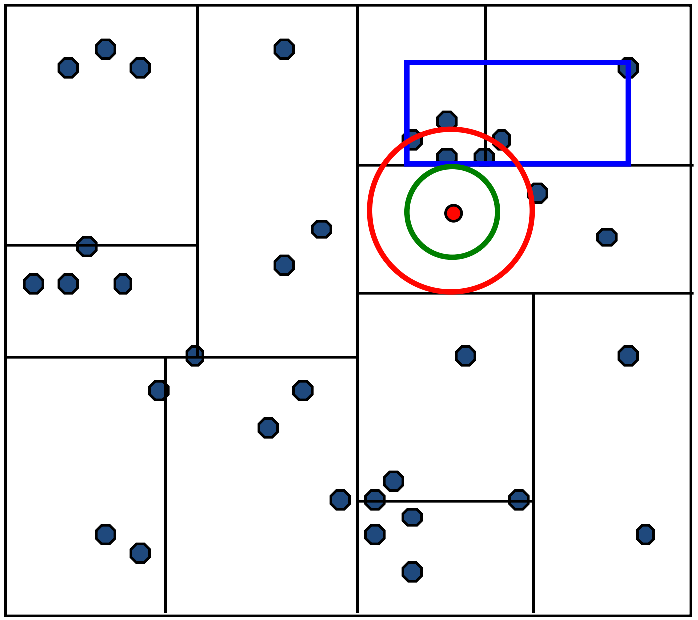
K-NN with KD Trees

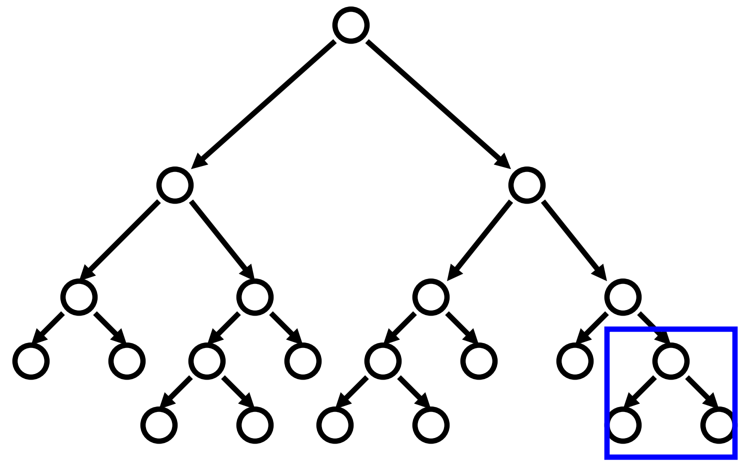




- Exactly the same algorithm, but maintain distance as distance to furthest of current *k* nearest neighbors
- Complexity is:

Approximate K-NN with KD Trees



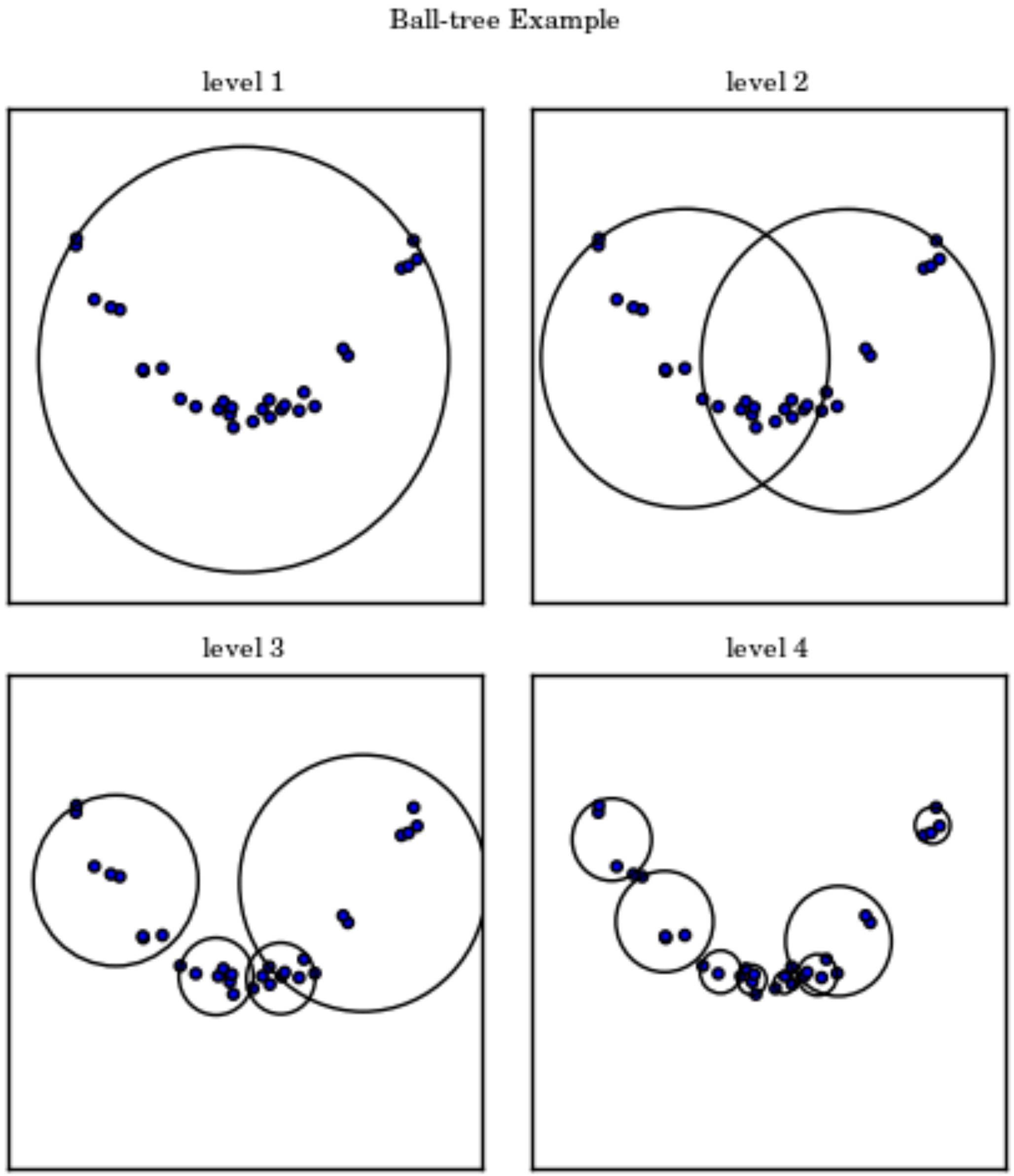


- **Before:** Prune when distance to bounding box >
- **Now:** Prune when distance to bounding box >
- Will prune more than allowed, but can guarantee that if we return a neighbor at distance r, then there is no neighbor closer than r/α.
- In practice this bound is loose…Can be closer to optimal.
- Saves lots of search time at little cost in quality of nearest neighbor.

What about NNs searches in high dimensions?

- KD-trees:
  - What is going wrong?
  - Can this be easily fixed?
- What do have to utilize?
  - utilize triangle inequality of metric
  - New ideas: ball trees and cover trees

Ball Trees



Ball Tree Construction

- Node:
  - Every node defines a ball (hypersphere), containing
    - a subset of the the points (to be searched)
    - A center
    - A (tight) radius of the points
- Construction:
  - Root: start with a ball which contains all the data
  - take a ball and make two children (nodes) as follows:
    - Make two spheres, assign each point (in the parent sphere) to its closer sphere
    - Make the two spheres in a “reasonable” manner

Ball Tree Search

- Given point x, how do find its nearest neighbor quickly?
- Approach:
  - Start: follow a greedy path through the tree
  - Backtrack and prune: rule out other paths based on the triange inequality
    - (just like in KD-trees)

- How good is it?
  - Guarantees:
  - Practice:

Cover trees

- What about exact NNs in general metric spaces?
- Same Idea: utilize triangle inequality of metric (so allow for arbitrary metric)
- What does the dimension even mean?
- cover-tree idea:

Intrinsic Dimension

- How does the volume grow, from radius R to 2R?

- Can we relax this idea to get at the "intrinsic" dimension?

  - This is the "doubling" dimension:

NN complexities

	Query time	Space used	Preprocessing time
Vornoi	$O\left(2^d \log n\right)$	$O\left(n^{d/2}\right)$	$O\left(n^{d/2}\right)$
Kd-tree	$O\left(2^d \log n\right)$	$O(n)$	$O(n \log n)$
LSH	$O\left(n^{\rho} \log n\right)$	$O\left(n^{1+\rho}\right)$	$O\left(n^{1+\rho} \log n\right)$