ONLINE MASTERS IN **DATA SCIENCE**

DSC 255 - MACHINE LEARNING FUNDAMENTALS

SPEEDING UP NEAREST NEIGHBOR METHODS

SANJOY DASGUPTA, PROFESSOR

UC San Diego

COMPUTER SCIENCE & ENGINEERING

HALICIOĞLU DATA SCIENCE INSTITUTE

Algorithmic issue: speeding up NN search

Naive search takes time *O(n)* for training set of size *n*: slow!

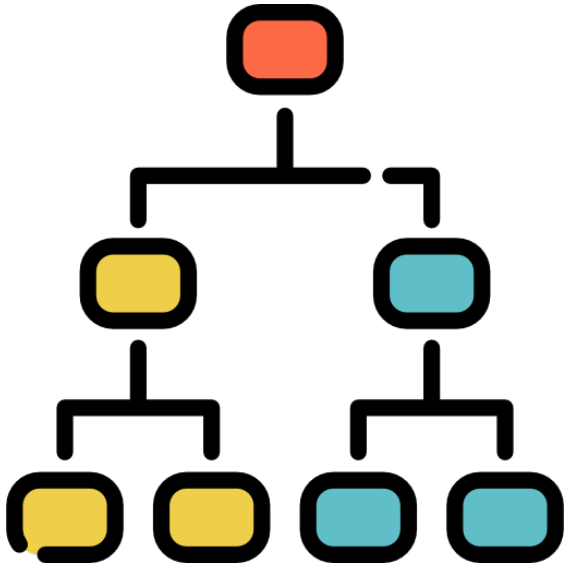
Algorithmic issue: speeding up NN search

Naive search takes time *O(n)* for training set of size *n*: slow!

There are data structures for speeding up nearest neighbor search, like:

1. Locality sensitive hashing
2. Ball trees
3. *K*-d trees

These are part of standard Python libraries for NN, and help a lot.

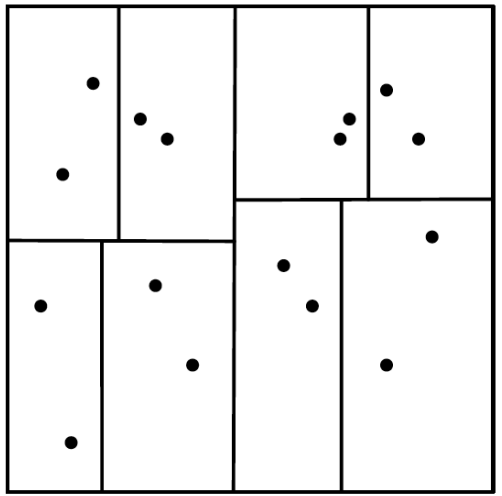


Example: *k*-d trees for NN search

hierarchical, rectilinear spatial partition.

For data set $S \subset \mathbb{R}^d$:

1. Pick a coordinate $1 \leq i \leq d$.
2. Compute $\mathbf{v = median}(\{x_i : x \in S\})$.
3. Split S into two halves:
   - $S_L = \{x \in S : x_i < v\}$
   - $S_R = \{x \in S : x_i \geq v\}$
4. Recurse on S_L, S_R

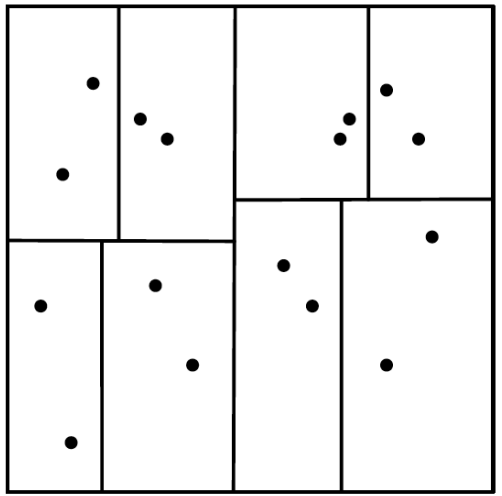


Example: *k*-d trees for NN search

hierarchical, rectilinear spatial partition.

For data set $S \subset \mathbb{R}^d$:

1. Pick a coordinate $1 \leq i \leq d$.
2. Compute $\mathbf{v = median}(\{x_i : x \in S\})$.
3. Split S into two halves:
   - $S_L = \{x \in S : x_i < v\}$
   - $S_R = \{x \in S : x_i \geq v\}$
4. Recurse on S_L, S_R



Two types of search, given a query $q \in \mathbb{R}^d$:

1. **Defeatist search:** Route q to a leaf cell and return the NN in that cell. This might not be the true NN.
2. **Comprehensive search:** Grow the search region to other cells that cannot be ruled out using the triangle inequality.