Five flavors of "nearest-neighbor"

1. Exact NN search
2. Approximate NN search
3. Known-query NN lookup
4. All-queries NN search
5. NN classification

Exact NN search

- *kd*-trees (MR*kd*-trees in our case)
  - Disk-resident: R-trees, SR-trees
- Metric trees (aka ball-trees)
- Other well-known trees
  - 'One-sided' ball-trees, e.g. VP-trees
  - PCA trees, e.g. Sproull's trees
  - many more, esp. minor variants

Approximate NN search

Idea: Don't return exact NN; return a point whose distance is $(1+\varepsilon)$ times the true NN distance. e.g.:

- BBD-tree
- Clarkson's algorithm
- Locality-sensitive hashing (LSH)
- Overlap tree

Caveat: Not a guarantee on the actual rank!

Known-query NN lookup

Constraint: Only known points (reference points) are allowed as queries. e.g.:

- Planar separator/geometric separator
- Navigating nets
- Cover trees (coming soon)

Caveat: Typically requires $O(N^2)$ up-front cost!

All-queries NN search

Idea: If a whole set of queries is available (vs. one by one), more efficiency possible.

Two cases: 1) Known-query case ("all-NN"), 2) Arbitrary-query case ("bichromatic") e.g:

- Rabin's algorithm (all-NN case)
- WSPD (all-NN case)
- Dual-tree algorithm (bichromatic)

Caveat: Time to build data structure for query set must be counted.

NN classification

Idea: To classify, we don't need to find the actual nearest neighbors for each class. We just need to know which class the majority label is from.

- Two-priority-queue algorithm
- IOC (generalization to *k* classes)

Caveat: You only get the (exact) class prediction.