CSEP 521: Applied Algorithms
Lecture 14 – Nearest neighbors

Richard Anderson

February 18, 2021

Announcements

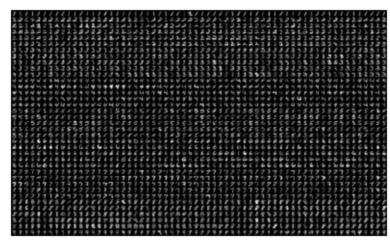
- Homework schedule
  - Homework 7, Due Thursday, February 25, 11:59 pm.
  - Homework 8, Due Thursday, March 4, 11:59 pm.
  - Homework 9, Due Thursday, March 11, 11:59 pm.
  - ~~Homework 10, Due Thursday, March 18, 11:59 pm.~~

High dimensional searching

- Many data sets are high dimensional
  - High dimension can mean a mathematical space, such as R^d, or a structure, such as bag-of-words representation of documents
- Canonical problem:
  - Given a new datum x, find the closest element y in the dataset
- Lots of things need to be defined, like "closest"
- Think of the data set as being very large, so we would like a mechanism that avoids having to do comparisons with all elements

Nearest neighbor motivation

Find closest match to a query in a large data set



Outline

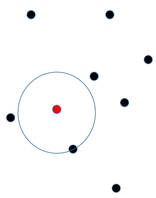
- Metric (distance measures)
- Coding theory
- Searching in 2-d
  - Quad trees
  - Voronoi diagrams
- Higher (but not too high) dimensions
  - K-d trees

Concepts

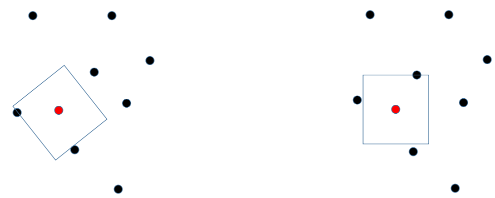
- Metric
  - Distance measure, d(x,y), d: $A \times A \rightarrow [0, \infty)$
  - Properties
    - d(x,y) = 0 iff x = y
    - d(x,y) = d(y,x)
    - d(x,y) ≤ d(x,z) + d(z,y)
    - d(x,y) ≥ 0
- Standard Euclidean distance – L^2 Norm $\|(x,y)\|_2 = \sqrt{x^2+y^2}$
- L^p Norm $\|(x,y)\|_p = \sqrt[p]{x^p+y^p}$
- L^1 Norm $\|(x,y)\|_1 = |x+y|$
- L^∞ Norm $\|(x,y)\|_\infty = \max(x,y)$

Nearest neighbor problem

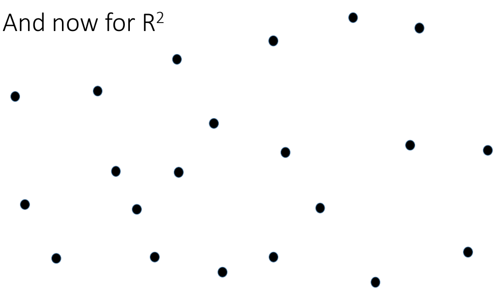
- Set of points S
- Given query point y, find a point in S closest to y



Nearest neighbor problem: L_1 and L_∞ metrics

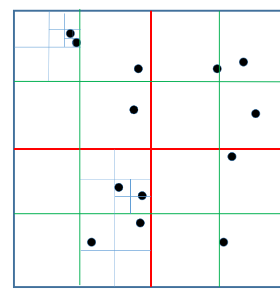


And now for R^2



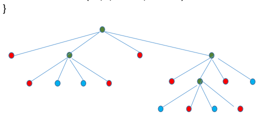
Quad Tree

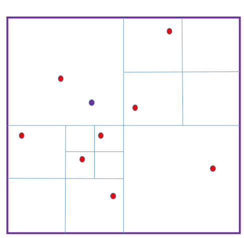
- Start with a bounding square
- Each level divides a square into four quadrants
- Search explores cells which may contain nearest neighbor
  - Track best-so-far distance to prune sub trees in recursive tree traversal
- Depth is determined by closest pair distance



Nearest Neighbor Search

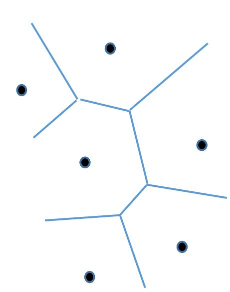
```
Search(tree T,  point P,  int bound, point closest){
    if  leaf node
        if non-empty
            if (dist(P, X) < bound)  update bound and closest
    else
        foreach subtree T1
            if (dist(P, T.Region) < bound)
                Search(T1, P, bound, closest)
}
```



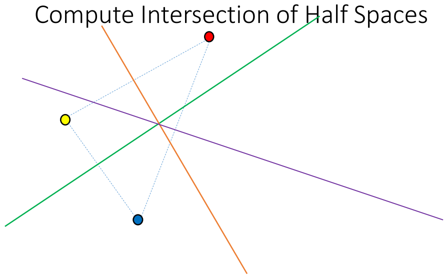


Voronoi diagram

- For each point x, Voronoi region is the set of points (in R^2) where x is the nearest neighbor in S
- Between each pair of points we can look at the separating half spaces
- A points Voronoi region is the intersection of half spaces (and convex)
- The number of segments is O(N)

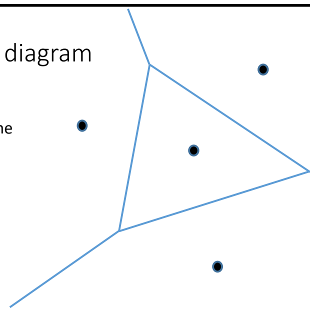


Voronoi Regions
Compute Intersection of Half Spaces



Building the Voronoi diagram

- Lots of algorithms exist
- It can be done in O(n log n) time
- Programming is a challenge
  - Lots of special cases
  - Careful numerical programming
  - Hard to debug
- Most practical algorithm is probably to insert points in random order into an existing diagram



Search in a Voronoi diagram

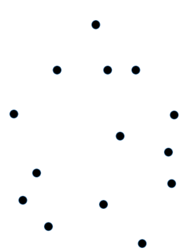
- Need to overlay a search structure on top of the diagram
- Can use a sequence of separating segments
- Binary space partition trees can be used
- In theory, this can be done in O(log n) query time

What about 3 dimensions?

- Quad trees generalize to oct-trees in 3d, with 8 children instead of 4
- Unfortunately, the 3-d Voronoi tessellation (honeycomb) can have size n^2
  - Proof: divide the points into to sets A and B, and put A and B on separate arcs. This can be done so that each point a_i in A shares a face with each b_j in B

K-D trees

- Another spatial decomposition tree
  - Bentley, 1975
- Separate across dimensions in order d_1, d_2, d_3, . . .
- Split point sets evenly, not space evenly



KD-Tree construction

- Find median point in dimension d_j
- Split points into left/right
- Recursively decompose regions
- Maintain bounding boxes and/or splitting axis
- Tree depth is O(log n)
- Tree construction is O(n log n)

Tree operations

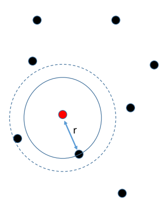
- Locate point
  - Traverse tree
- Range query: return points inside a bounding box
  - Traverse tree
- Nearest neighbor search
  - Traverse tree

Comparison between KD Trees and generalized Quad Trees

- KD trees have degree 2 and height log n
- Gen-Quad Trees have degree 2^d and height dependent on point distribution
- KD bounding boxes can be narrow
- Gen-Quad Trees are cubes
- KD trees generally preferred for $d \geq 4$

Approximate closest points

- Approximate closest point
  - Suppose the closest point distance from y to a point in S is r
  - Find a point in S that has distance $(1+\varepsilon)r$ from y



Approximate closest points

- Nearest neighbor search

```
if (dist(P, T.Region) < bound)
        Search(T1, P, bound, closest)
```

- Approximation algorithm

```
if ((1+ε)*dist(P, T.Region) < bound)
        Search(T1, P, bound, closest)
```