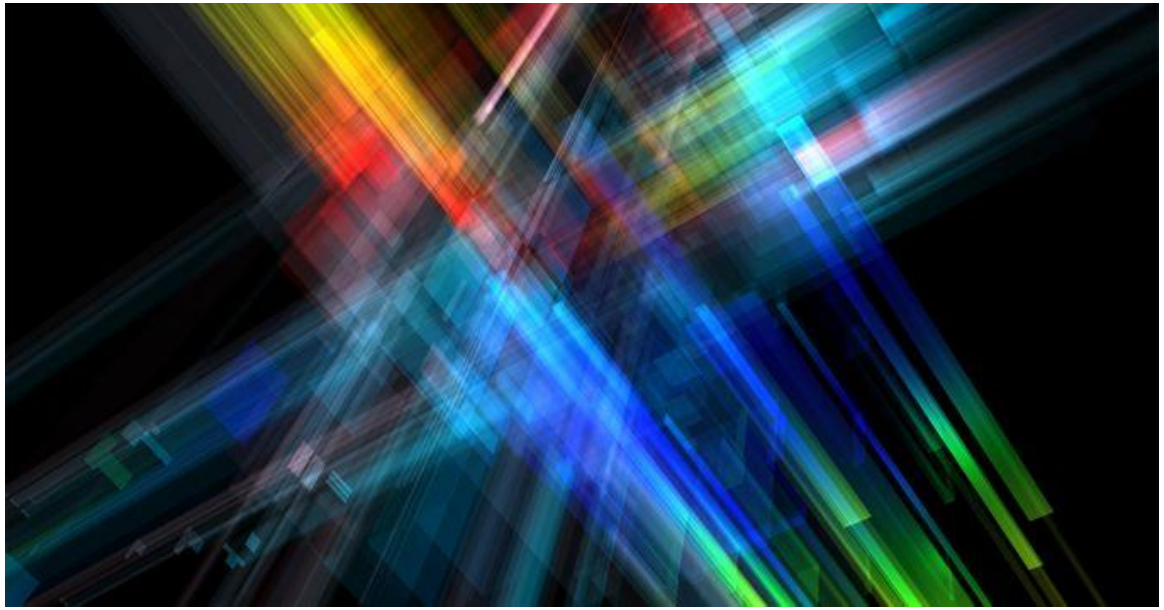
CSEP 521: Applied Algorithms
Lecture 13 – Geometry and Searching

Richard Anderson

February 16, 2021



Announcements

-

Course outline

- Probabilistic algorithms and average case analysis
- Sublinear space algorithms for streaming
- Geometry and searching
  - Nearest neighbor problems
  - Low dimensional searching
  - Higher dimensions
  - Locally sensitive hashing
  - Document similarity
  - Linear programming

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

Outline for this week

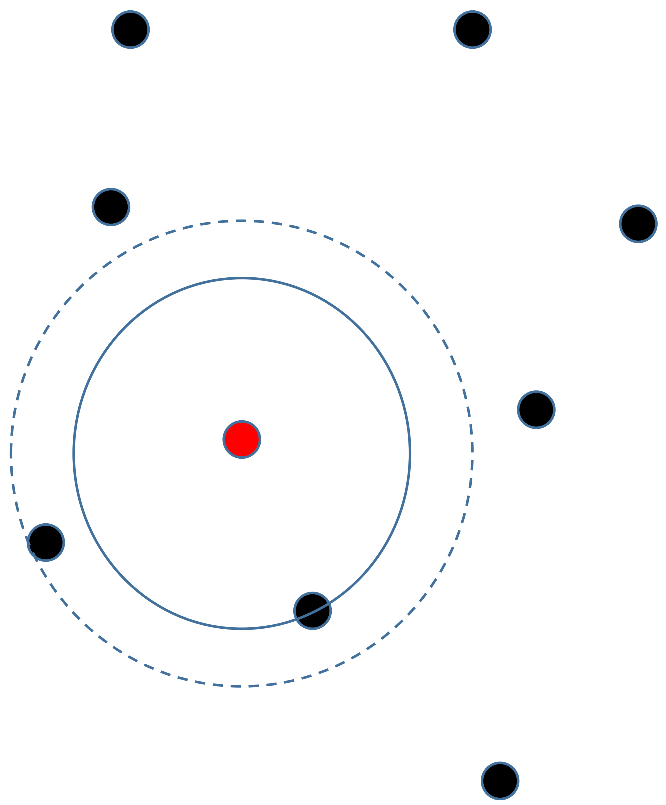
- Concepts
- Coding theory
- Searching in 2-d
  - Quad trees
  - Voronoi diagrams
- Higher (but not too high) dimensions
  - K-d trees

Concepts

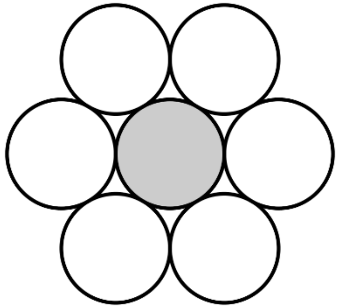
- Product space
  - Mathematically – Cartesian Product
  - Euclidean space, R^d
  - Other spaces, Z_p^d
- Metric
  - Distance measure, $d(x,y)$, $d: A \times A \to [0, \infty)$
  - Properties
    - $d(x,y) = 0$ iff $x = y$
    - $d(x,y) = d(y,x)$
    - $d(x,y) \leq d(x,z) + d(z,y)$
    - $d(x,y) \geq 0$

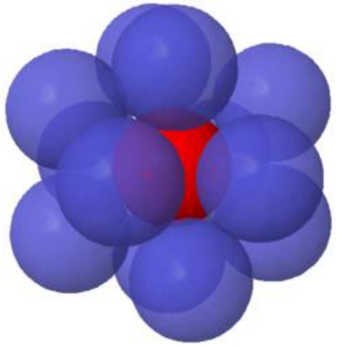
Closest points and approximate closest points

- Set of points S
- Given query point y, find a point in S closest to y
- ε - 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+ε)r from y



Intuition and where it breaks down





- My pictures are in R^2
- I can imagine what happens in R^3
- Higher dimensions are much, much harder
- Imagine an N dimensional sphere

- Low dimensional intuition is not necessarily good for higher dimensions
  - Many quantities grow exponentially with dimension
  - Kissing number – number n dimensional sphere that can be arranged to touch a single sphere

Dim	Lower	Upper
2	6	6
3	12	12
4	24	24
8	240	240
12	840	1357
16	4320	3183
20	17400	36764
24	196560	196560
28	204368	
32	276032	
36	484568	

Warm up – Coding theory

- Problem – sending data across a noisy channel

- Model
  - Words encoded as a block of digits
  - Digits are transmitted
  - Some digits may be changed
  - The block is decoded to get the message

Idea one – parity bit

- Add a parity bit or check sum
  - Message $x_1, x_2, \ldots, x_k$
  - Let $y = x_1 \oplus x_2 \oplus \ldots \oplus x_k$ (exclusive OR)
  - Send message $x_1, x_2, \ldots, x_k, y$
- Resulting message has even parity
- If a block is received with odd parity, at least one bit was flipped
- Single error detection

Idea two - redundancy

- Make three copies of each code word
- One error correcting
- Every pair of code words is at distance at least three
- But this is a very dumb code

Text	Codeword
000	000000000
001	001001001
010	010010010
011	011011011
100	100100100
101	101101101
110	110110110
111	111111111

Block codes

- Coding theory / Information theory started in 1940s at Bell Labs
  - Clause Shannon, Richard Hamming

- $(n,k,d)_q$: Alphabet size q (omit for 2), block length n, message length k, distance d
  - k/n gives the rate
- $(n,k,d)_q$ code can detect d-1 errors and correct $\lfloor (d-1)/2 \rfloor$ errors

Hamming(7,4) code

- Linear code with 3 parity bits
- Basis vectors
  - [1,1,1,0,0,0,0]
  - [1,0,0,1,1,0,0]
  - [0,1,0,1,0,1,0]
  - [1,1,0,1,0,0,1]
- Encoding / Decoding / Error correction are linear algebra operations over Z_2

Golay Code: G_{24} $[24,12,8]_2$ and G_{23} $[23,12,7]_2$

- Closely related codes, 12 dimensional subspaces of Z_2^{24} and Z_2^{23} respectively
- G_{24} is used because it is 3 bytes
- G_{23} is a perfect code. Spheres of radius 3 around the code words partition the vector space

- Imagine a 23 dimensional sphere of radius three centered at z, find the codeword in the sphere

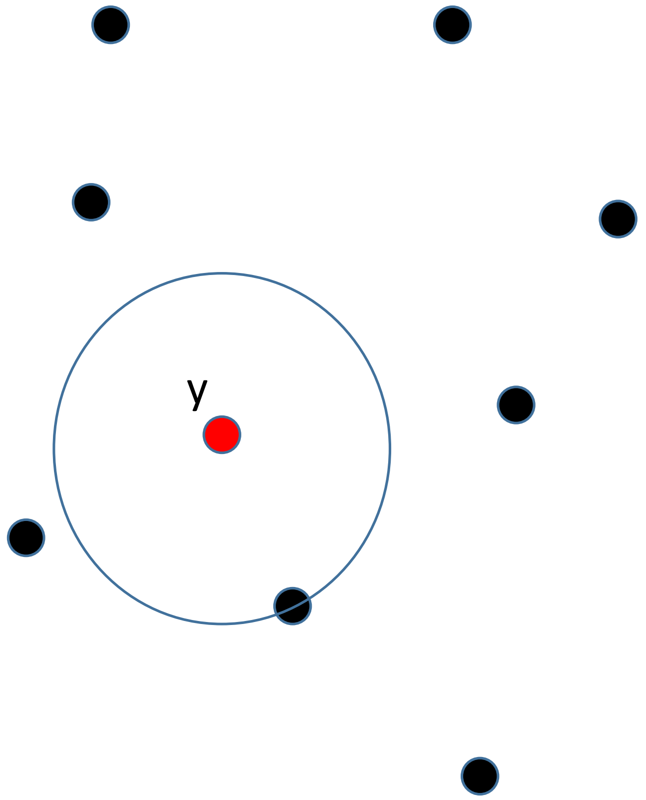
Hamming distance

- A and B are binary vectors of length k
- Hamming Distance, HD(A,B) is the number of positions where they differ
    - A = 10110010
    - B = 00100110
    - HD(A,B) = 3

- Ball of radius r centered at A is the set of all B with HD(A,B)≤r
- A = 10011, r = 2, ball of radius 2: {10011, 00011, 11011, 10111, 10001, 10010, 01011, 00111, 00001, 00010, 11111, 11001, 11010, 10101, 10110, 10000}

Low dimensional problems

- $S = \{x_1, x_2, . . . , x_n\}$
- Given an value y, find the closest point in S to y.
  - $\text{Min}_i \; d(y, x_i)$

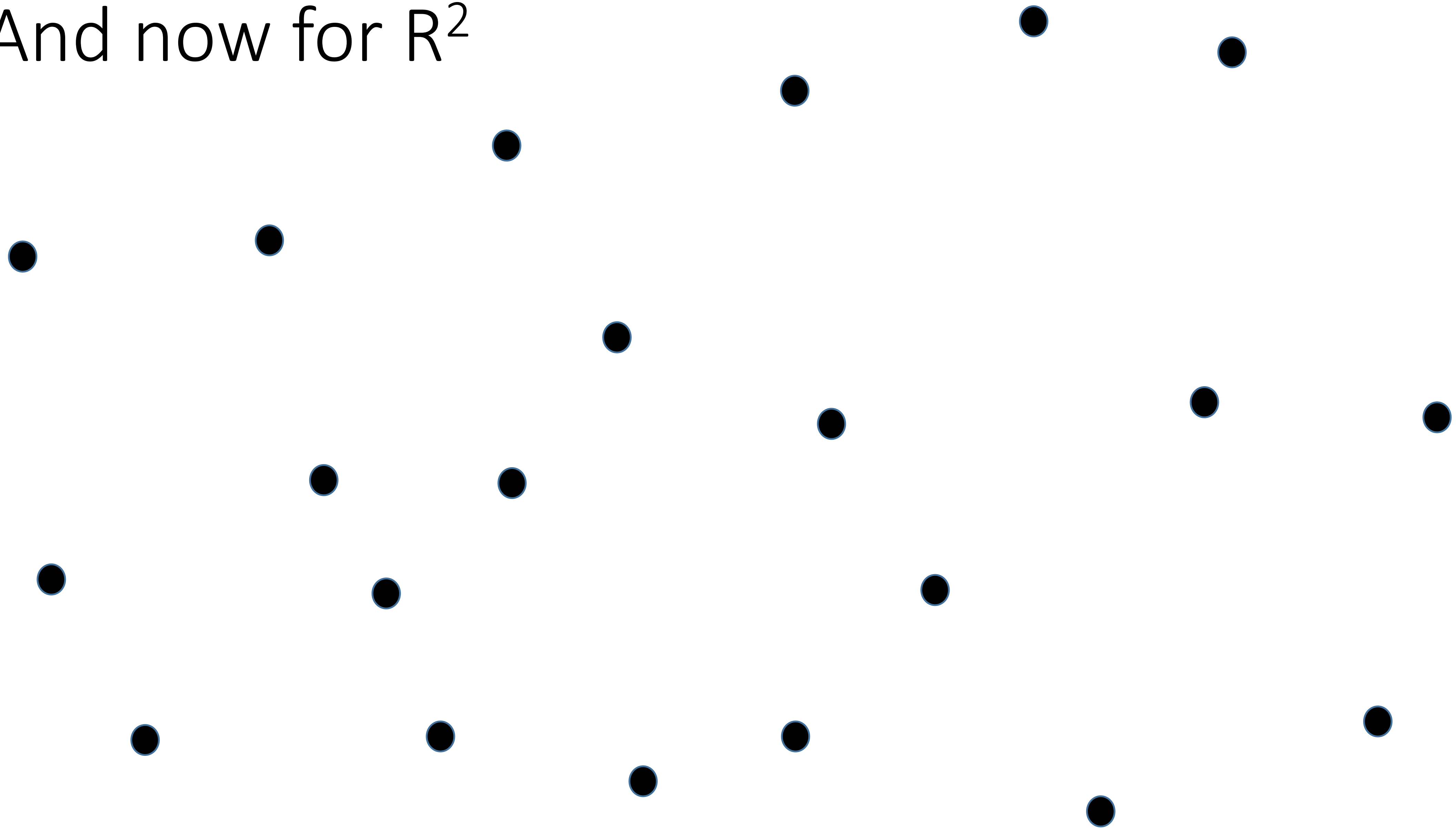


How do we solve this in R^1

Issues

- Static versus dynamic data structures
- Average case versus worst case
- Numerical precision of coordinates

And now for R^2



What is the distance function?

- 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

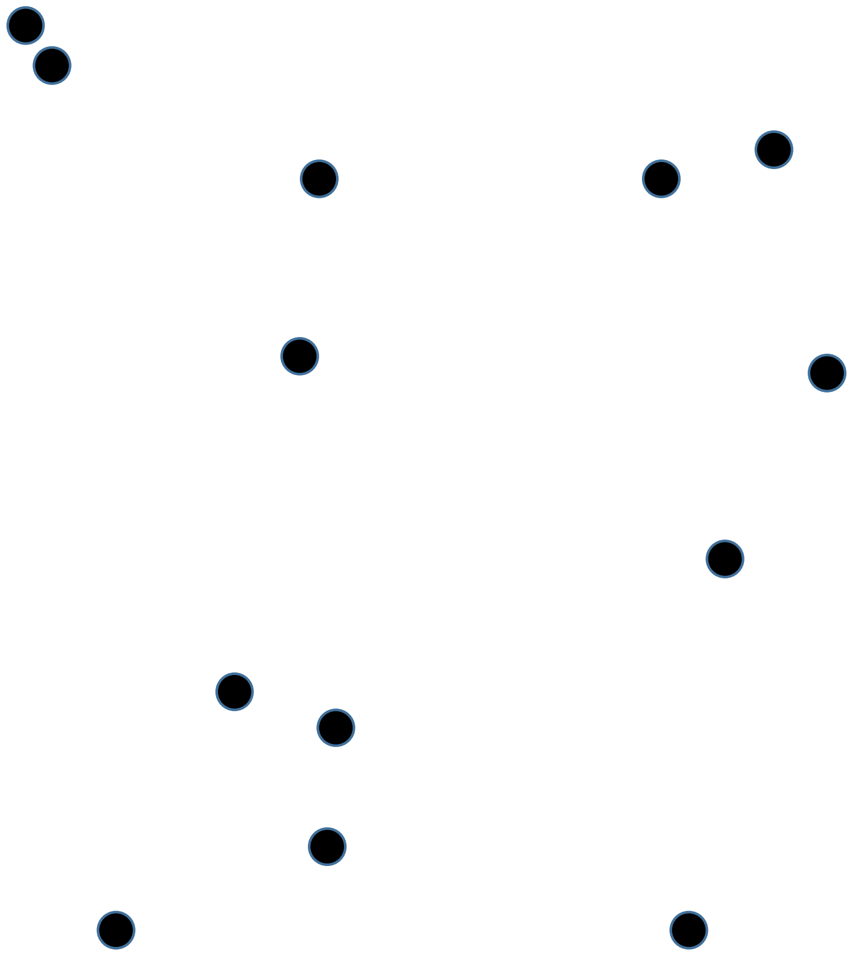
$$\|(x, y)\|_\infty = \max(x, y)$$

Data structures for 2-d nearest neighbor

- Unlike 1-d we do not have a linear order on the points
- Multiple options are available (and variants exist)
  - Quad trees
  - K-d trees
  - Voronoi diagram

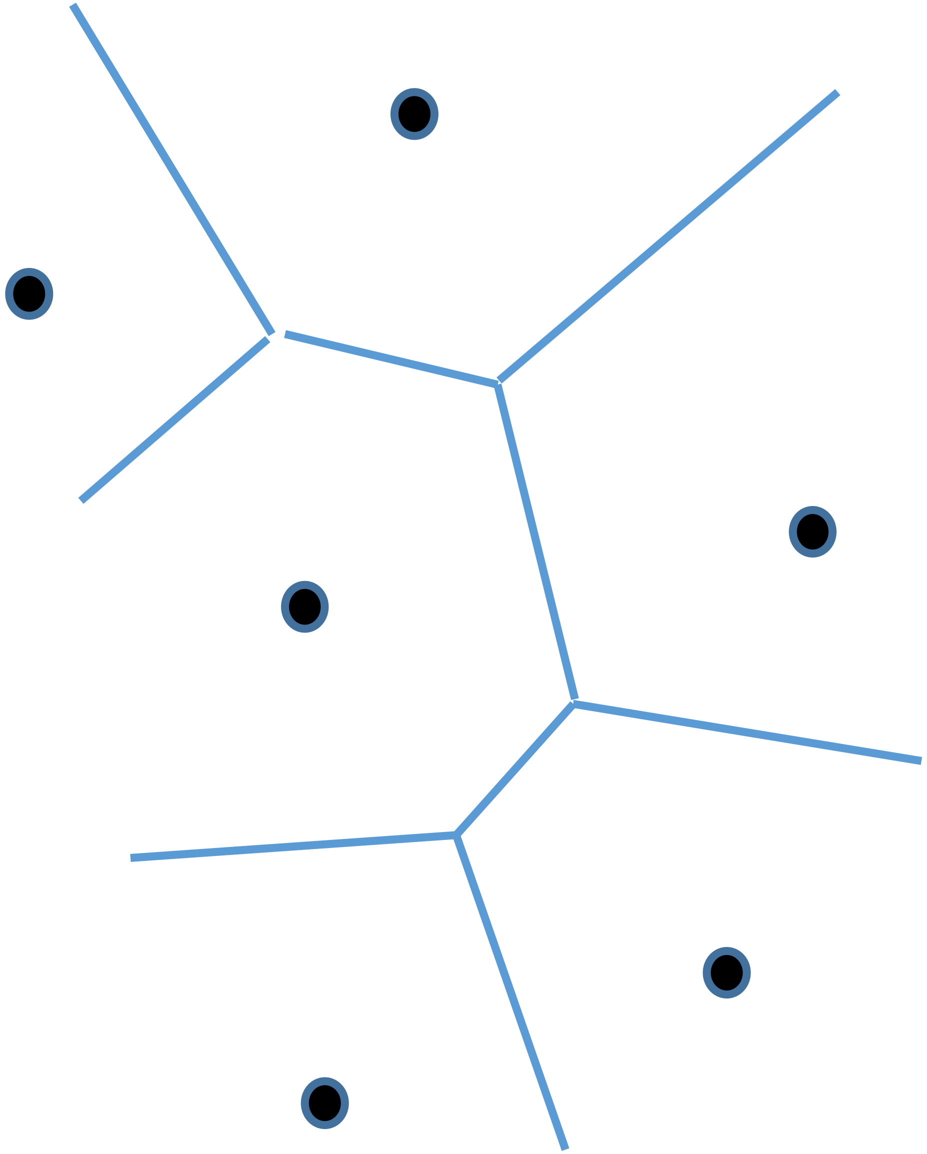
Quad Tree

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



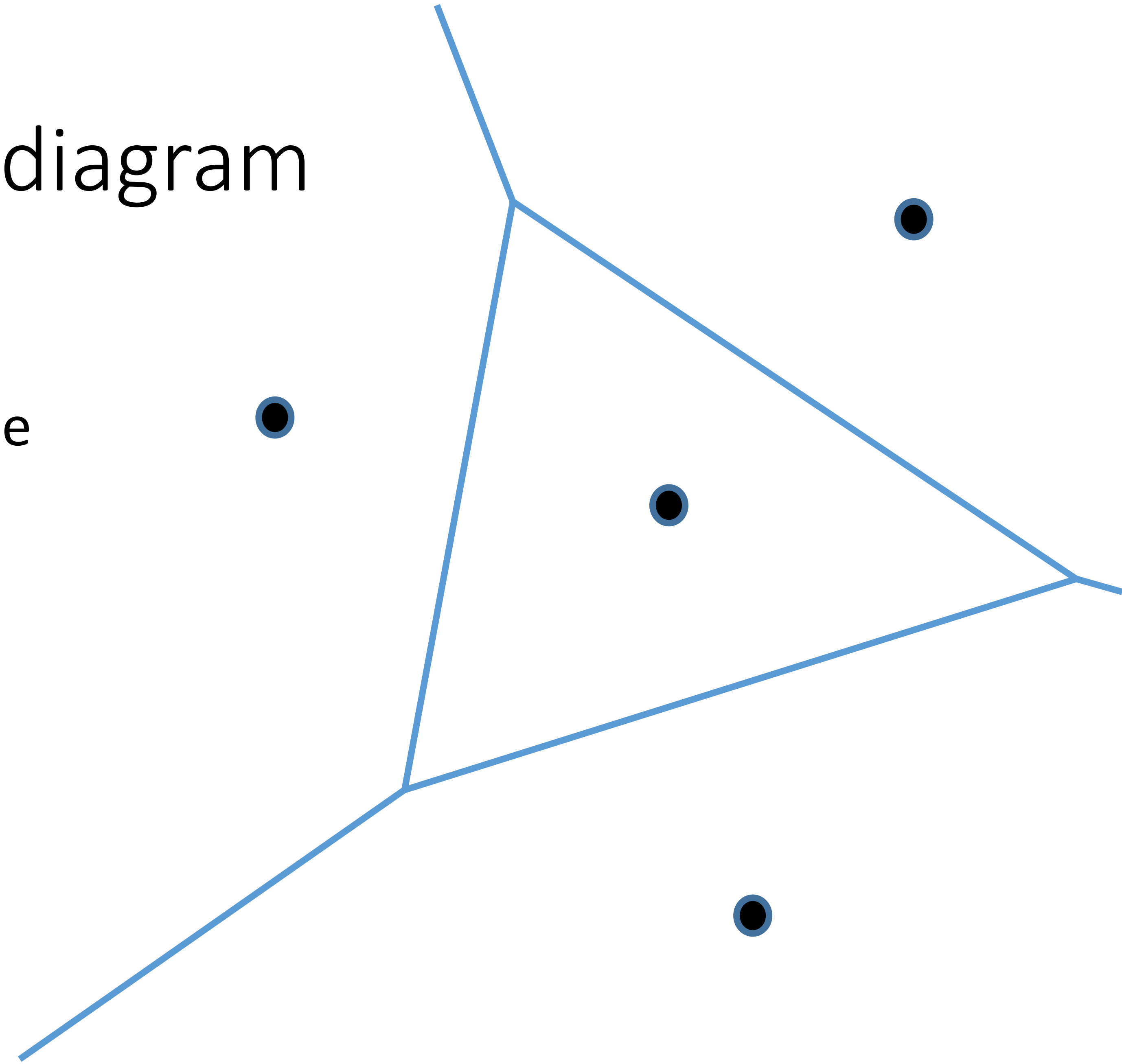
Voronoi diagram

- For each point x in S, the 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 point's Voronoi region is the intersection of half spaces (and is convex)
- The number of segments is O(N)



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 a Voronoi 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 3-d, 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