Lecture 12:

More LSH
Data-dependent hashing

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Announcements & Plan

- PS3:
  - Released tonight, due next Fri 7pm
- Class projects: think of teams!
- I'm away until Wed
  - Office hours on Thu after class
- Kevin will teach on Tue
- Evaluation on courseworks next week

- LSH: better space
- Data-dependent hashing
  - Scriber?

Time-Space Trade-offs (Euclidean)

space	query time	Space	Time	Comment	Reference
low	high	$\approx n$	n^{σ}	$\sigma = 2.09/c$	[Ind'01, Pan'06]
				$\sigma = O(1/c^2)$	[AI'06]
medium	medium	$n^{1+\rho}$	n^{ρ}	$\rho = 1/c$	[IM'98, DIIM'04]
				$\rho = 1/c^2$	[AI'06]
				$\rho \geq 1/c^2$	[MNP'06, OWZ'11]
		$n^{1+o(1/c^2)}$	ω(1) memory lookups		[PTW'08, PTW'10]
high	low	n^{4/ϵ^2}	$O(\cancel{d} \log n)$ 1 mem lookup	$c = 1+\epsilon$	[KOR'98, IM'98, Pan'06]
		$n^{o(1/\epsilon^2)}$	ω(1) memory lookups		[AIP'06]

Near-linear Space for $\{0,1\}^d$

[Indyk'01, Panigrahy'06]

Sample a few buckets in the same hash table!

- Setting:
  - Close: $r = \frac{d}{2c} \Rightarrow P_1 = 1 - \frac{1}{2c}$
  - Far: $cr = \frac{d}{2} \Rightarrow P_2 = \frac{1}{2}$
- Algorithm:
  - Use one hash table with $k = \frac{\log n}{\log 1/P_2} = \alpha \cdot \ln n$
  - On query q:
    - compute $w = g(q) \in \{0,1\}^k$
    - Repeat $R = n^\sigma$ times:
      - w': flip each w_j with probability $1 - P_1$
      - look up bucket $g(w')$ and compute distance to all points there
    - If found an approximate near neighbor, stop

Near-linear Space

- Theorem: for $\sigma = \Theta\left(\frac{\log c}{c}\right)$, we have:
  - Pr[find an approx near neighbor]≥ 0.1
  - Expected runtime: $O(n^{\sigma})$
- Proof:
  - Let p^* be the near neighbor: $||q - p^*|| \leq r$
  - $w = g(q)$, $t = ||w - g(p^*)||_1$
  - Claim 1: $\Pr_g\left[t \leq \frac{k}{c}\right] \geq \frac{1}{2}$
  - Claim 2: $\Pr_{g,w'}\left[w' = g(p) \,\middle|\, ||q - p||_1 \geq \frac{d}{2}\right] \leq \frac{1}{n}$
  - Claim 3: $\Pr[w' = g(p^*) \mid Claim\ 1] \geq 2n^{-\sigma}$
  - If $w' = g(p^*)$ at least for one w', we are guaranteed to output either p^* or an approx. near neighbor

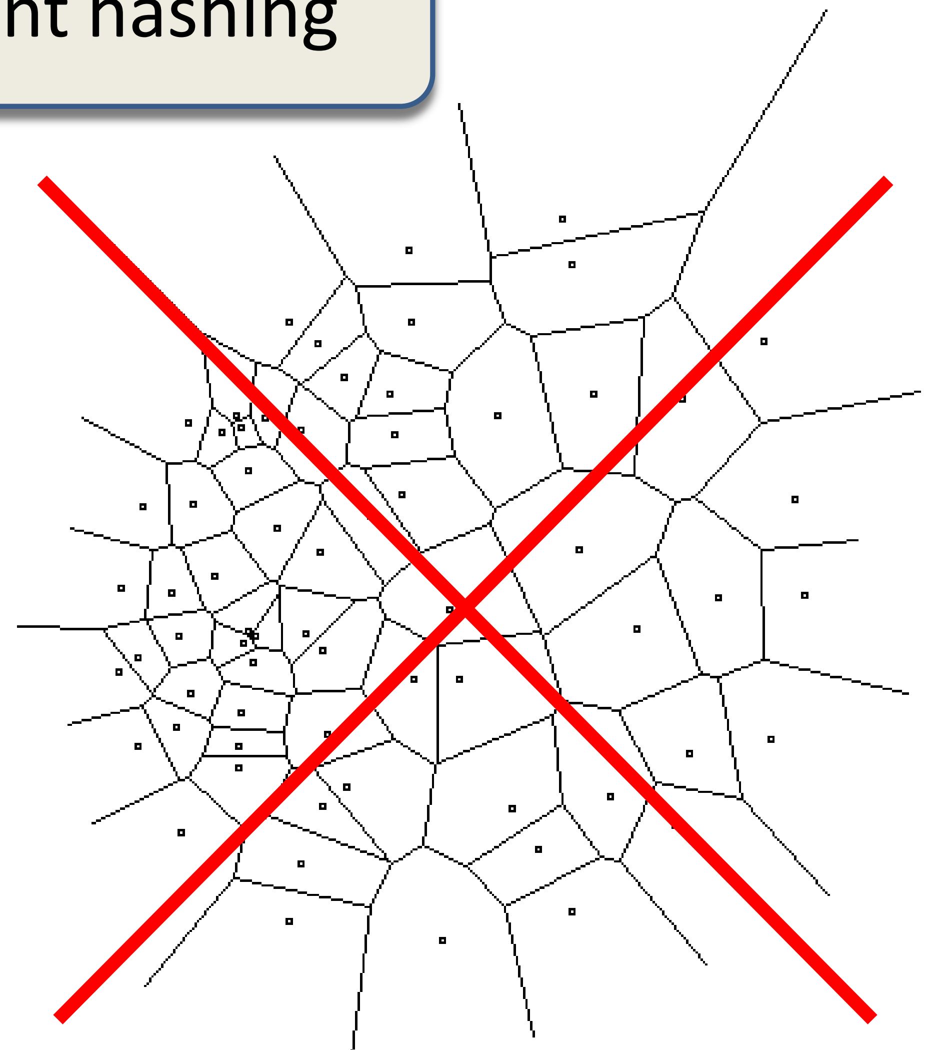
Beyond LSH

	Space	Time	Exponent	$c = 2$	Reference	
Hamming space	$n^{1+\rho}$	n^{ρ}	$\rho = 1/c$	$\rho = \mathbf{1/2}$	[IM'98]	LSH
			$\rho \geq 1/c$		[MNP'06, OWZ'11]	
	$n^{1+\rho}$	n^{ρ}	$\rho \approx \frac{1}{2c-1}$	$\rho = \mathbf{1/3}$	[AINR'14, AR'15]	
Euclidean space	$n^{1+\rho}$	n^{ρ}	$\rho \approx 1/c^2$	$\rho = \mathbf{1/4}$	[AI'06]	LSH
			$\rho \geq 1/c^2$		[MNP'06, OWZ'11]	
	$n^{1+\rho}$	n^{ρ}	$\rho \approx \frac{1}{2c^2-1}$	$\rho = \mathbf{1/7}$	[AINR'14, AR'15]	

New approach?

Data-dependent hashing

- A random hash function, chosen after seeing the given dataset
- Efficiently computable



Construction of hash function

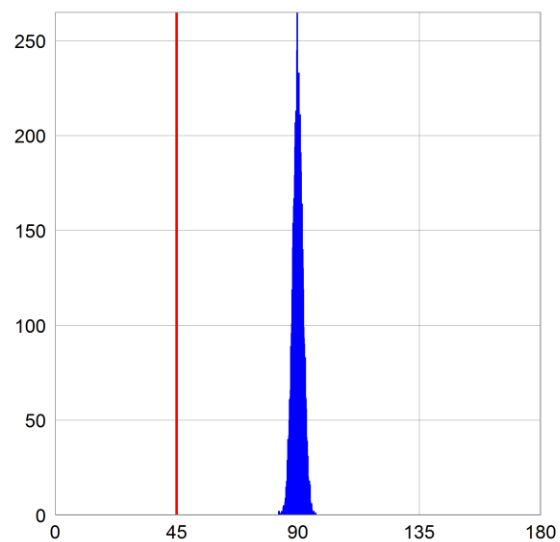
[A.-Indyk-Nguyen-Razenshteyn'14, A.-Razenshteyn'15]

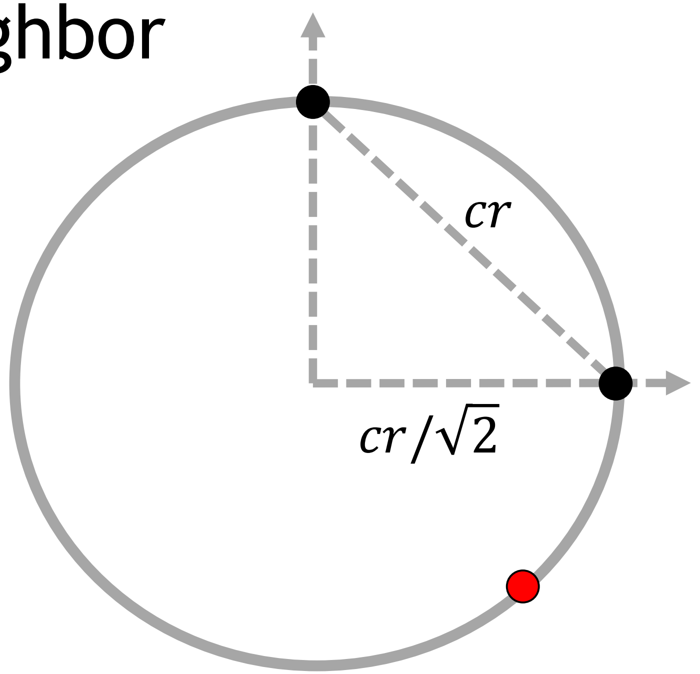
- Warning: hot off the press!

- Two components:
  - Nice geometric structure ← has better LSH
  - Reduction to such structure ← data-dependent

Nice geometric structure

- Like a random dataset on a sphere
  - s.t. random points at distance $\approx cr$
- Query:
  - At angle 45' from near-neighbor

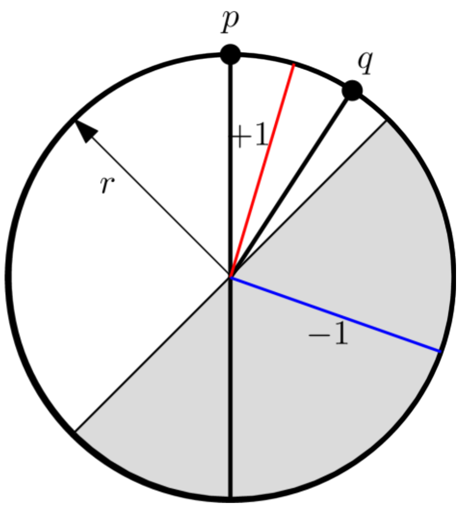




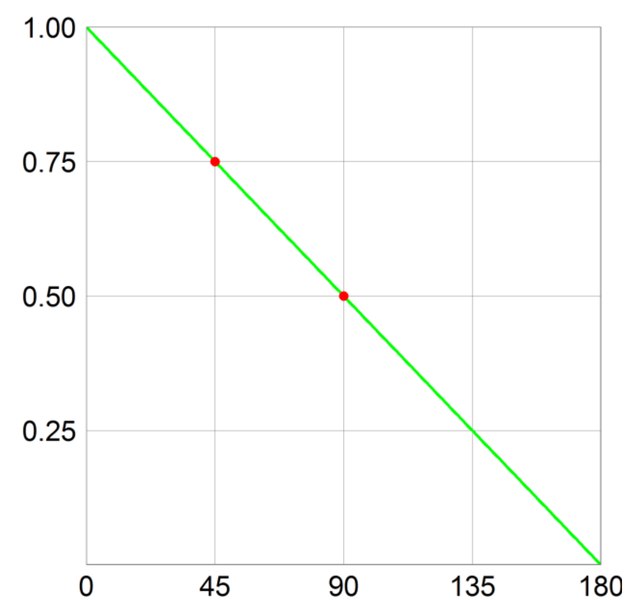
[Pictures curtesy of Ilya Razenshteyn]

Alg 1: Hyperplanes

[Charikar'02]



- **Sample *unit* r uniformly, hash p into $sgn\langle r, p\rangle$**
  - $\Pr[h(p) = h(q)] = 1 - \alpha / \pi$,
  - **where α is the angle between p and q**
- $P_1 = 3/4$
- $P_2 = 1/2$
- $\rho \approx 0.42$



Alg 2: Voronoi

[A.-Indyk-Nguyen-Razenshteyn'14] based on [Karger-Motwani-Sudan'94]

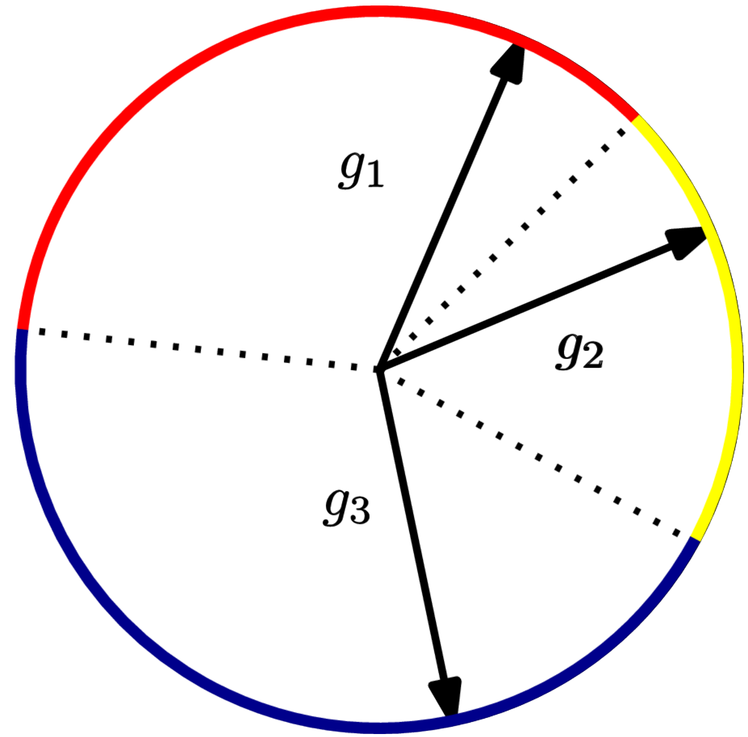
- **Sample T i.i.d. standard d-dimensional Gaussians**

$$g_1, g_2, \ldots, g_T$$

- **Hash p into**

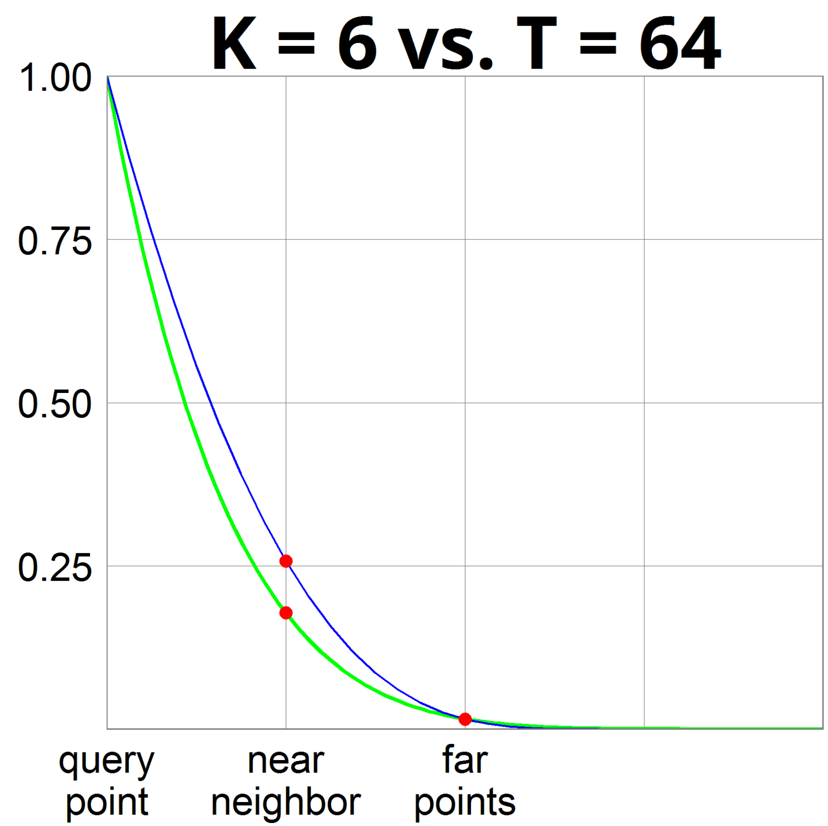
$$h(p) = \ argmax_{1 \le i \le T} \langle p, g_i \rangle$$

- **$T = 2$ is simply Hyperplane LSH**



Hyperplane vs Voronoi

- Hyperplane with $k = 6$ hyperplanes
  - Means we partition space into $2^6 = 64$ pieces
- Voronoi with $T = 2^k = 64$ vectors
  - $\rho = 0.18$
  - grids vs spheres



NNS: conclusion

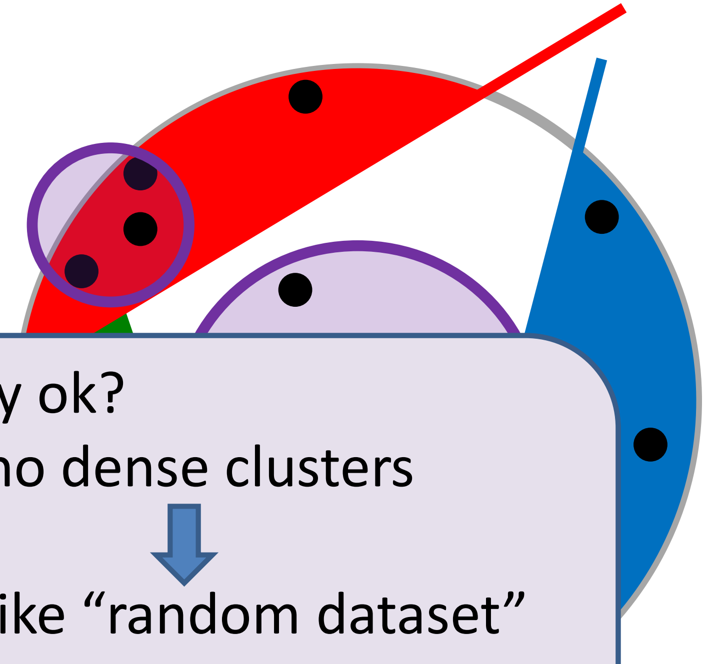
- 1. Via sketches
- 2. Locality Sensitive Hashing
  - Random space partitions
  - Better space bound
    - Even near-linear!
  - Data-dependent hashing even better
    - Used in practice a lot these days

- The following was not presented in the lecture

Reduction to nice structure (HL)

- Idea:
  iteratively decrease the radius of minimum enclosing ball

- Algorithm:
  - find dense clusters
    - with smaller radius
    - large fraction of points
  - recurse on dense clusters
  - apply VoronoiLSH on the rest
    - recurse on each "cap"
    - eg, dense clusters might reappear



Why ok?
- no dense clusters

⇩

- like "random dataset" with radius=$100cr$

⇩

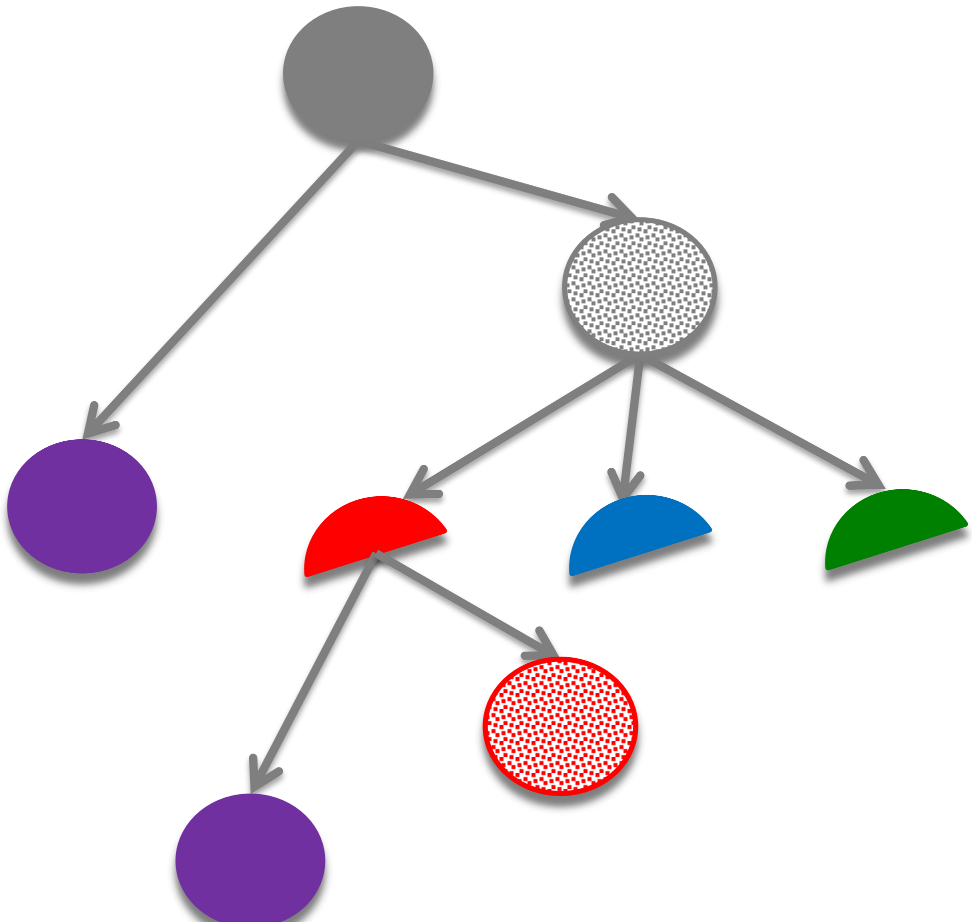
- even better!

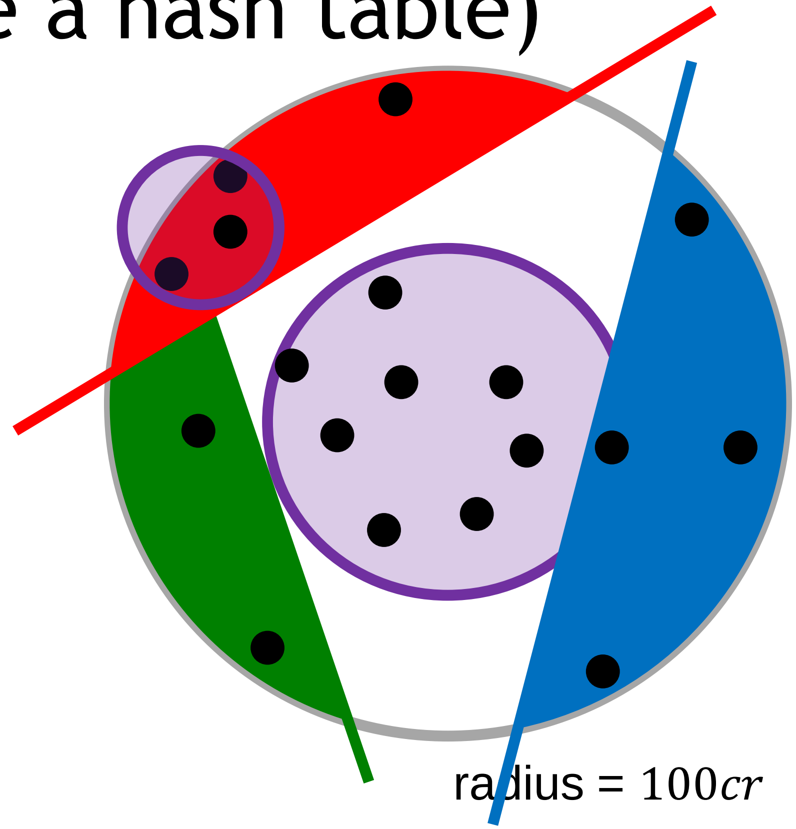
radius = $99cr$

*picture not to scale & dimension

Hash function

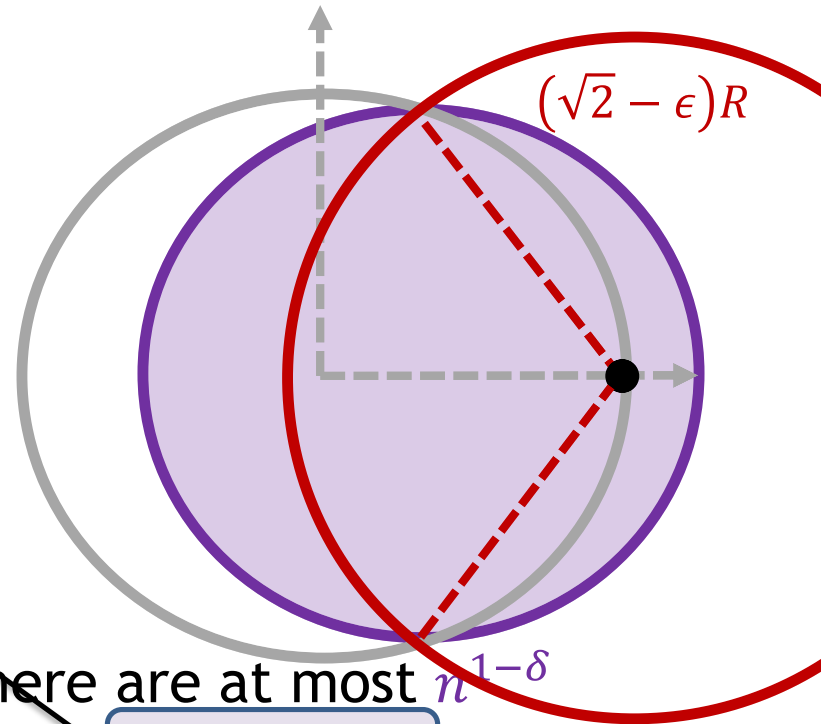
- Described by a tree (like a hash table)





*picture not to scale&dimension

Dense clusters



- Current dataset: radius R
- A dense cluster:
  - Contains $n^{1-\delta}$ points
  - Smaller radius: $(1-\Omega(\epsilon^2))R$
- After we remove all clusters:
  - For any point on the surface, there are at most $n^{1-\delta}$ points within distance $(\sqrt{2}-\epsilon)R$
  - The other points are essentially orthogonal !

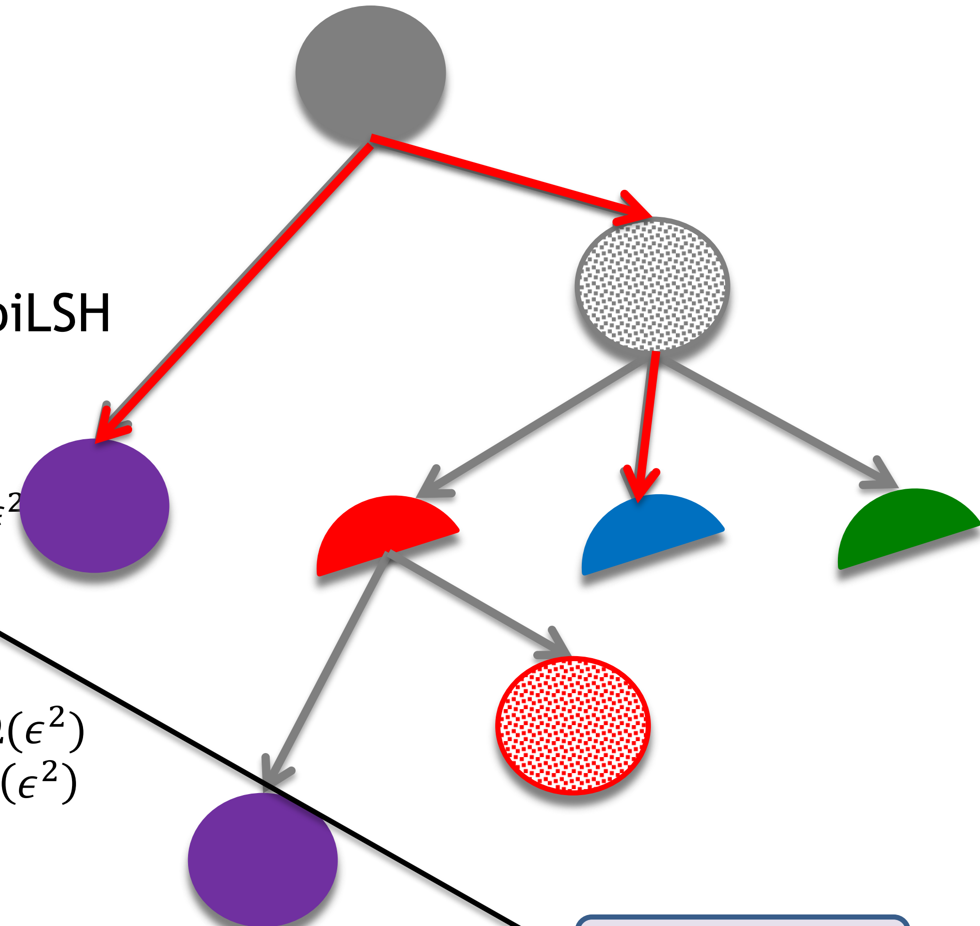
ϵ trade-off

- When applying Cap Carving with parameters (P_1, P_2):
  - Empirical number of far pts colliding with query: $nP_2 + n^{1-\delta}$
  - As long as $nP_2 \gg n^{1-\delta}$, the "impurity" doesn't matter!

δ trade-off

?

Tree recap

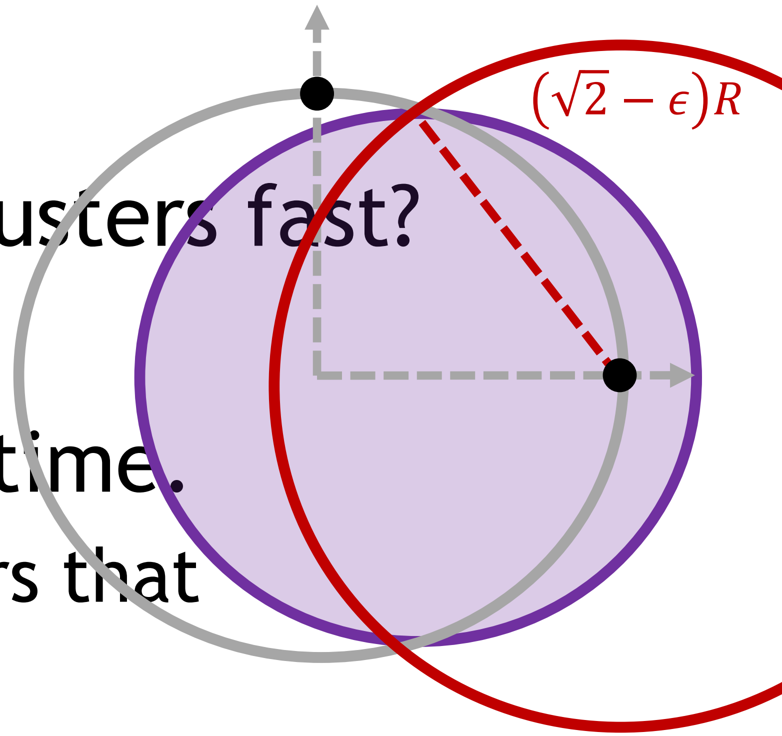


- During query:
  - Recurse in all clusters
  - Just in one bucket in VoronoiLSH
- Will look in >1 leaf!
- How much branching?
  - Claim: at most $(n^\delta + 1)^{O(1/\epsilon^2)}$
  - Each time we branch
    - at most n^δ clusters $(+1)$
    - a cluster reduces radius by $\Omega(\epsilon^2)$
    - cluster-depth at most $100/\Omega(\epsilon^2)$
- Progress in 2 ways:
  - Clusters reduce radius
  - CapCarving nodes reduce the # of far points (empiri

δ trade-off

- A tree succeeds with probability $\geq n^{-\frac{1}{2c^2-1}-o(1)}$

Fast preprocessing



- How to find the dense clusters fast?
- Step 1: reduce to $O(n^2)$ time.
  - Enough to consider centers that are data points
- Step 2: reduce to near-linear time.
  - Down-sample!
  - Ok because we want clusters of size $n^{1-\delta}$
  - After downsampling by a factor of $\sqrt{n}$, a cluster is still somewhat heavy.

Other details

- In the analysis,
  - Instead of working with “probability of collision with far point” P_2, work with “empirical estimate” (the actual number)
  - A little delicate: interplay with “probability of collision with close point”, P_1
    - The empirical P_2 important only for the bucket where the query falls into
    - Need to condition on collision with close point in the above empirical estimate
  - In dense clusters, points may appear *inside* the balls
    - whereas VoronoiLSH works for points on the sphere
    - need to partition balls into thin shells (introduces more branching)