Summer School on Hashing'14

Locality Sensitive Hashing

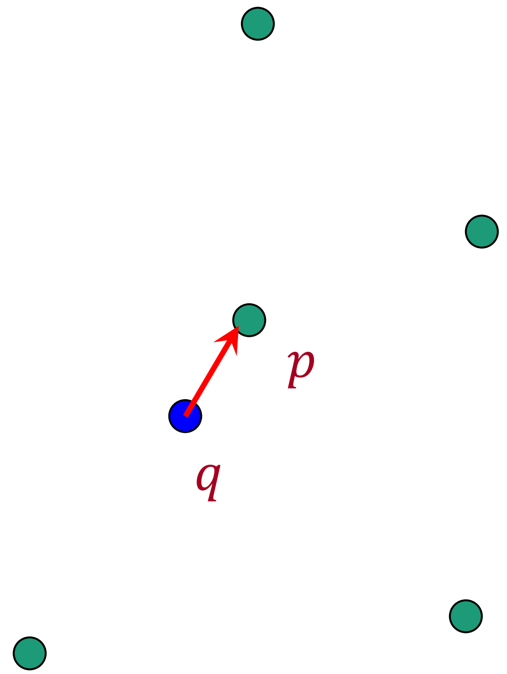
Alex Andoni

(Microsoft Research)

Nearest Neighbor Search (NNS)

- Preprocess: a set D of points

- Query: given a query point q, report a point $p \in D$ with the smallest distance to q



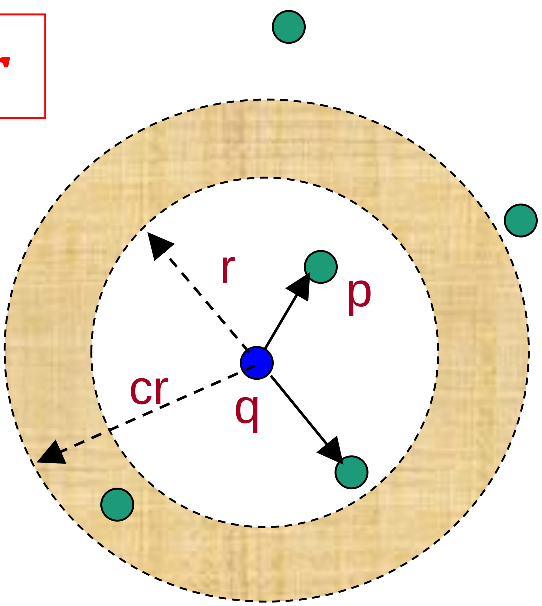
Approximate NNS

c-approximate

- r-near neighbor: given a new point q, report a point $p \in D$ s.t. $\|p - q\| \le \boxed{cr}$

  if there exists a point at distance $\le r$

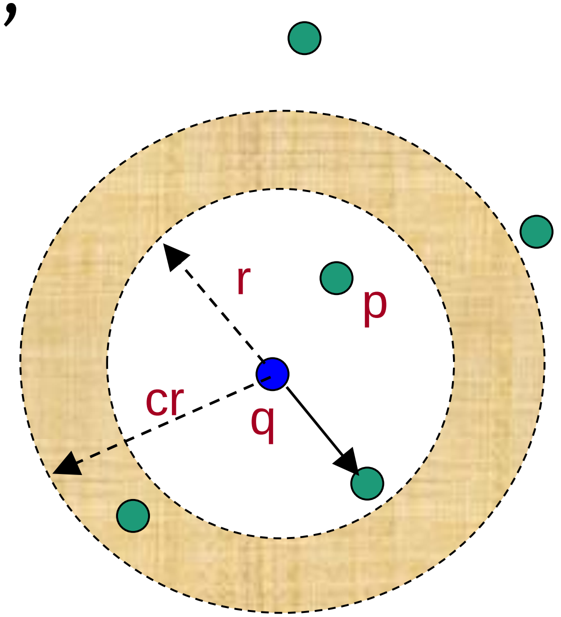
- Randomized: a point p returned with 90% probability



Heuristic for Exact NNS

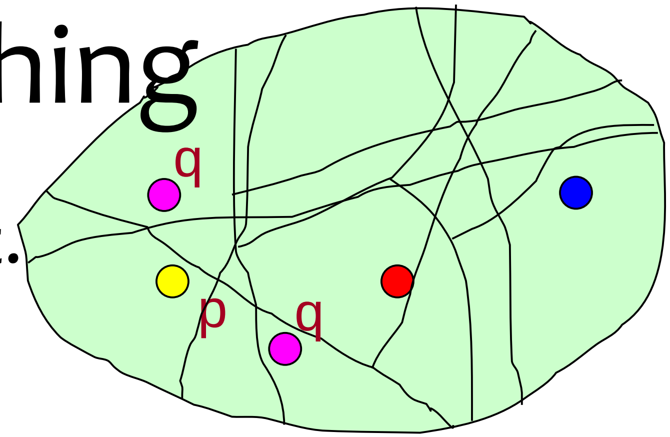
c-approximate

- r-near neighbor: given a new point q, report a set C with
  - all points p s.t. $\|p - q\| \leq r$ (each with 90% probability)
  - *may* contain some approximate neighbors p s.t. $\|p - q\| \leq cr$
- Can filter out bad answers



Locality-Sensitive Hashing

[Indyk-Motwani'98]

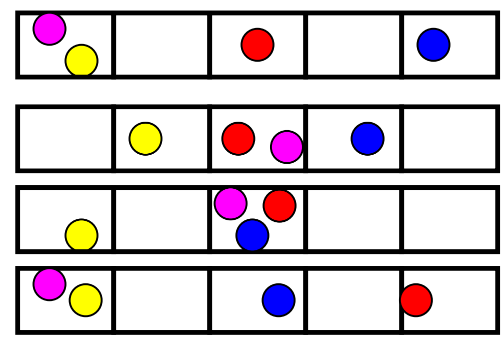


- Random hash function g on R^d s.t. for any points p, q:
  - *Close* when $\|p - q\| \le r$

$P_1 = \Pr[g(p) = g(q)]$ is "not-so-small"

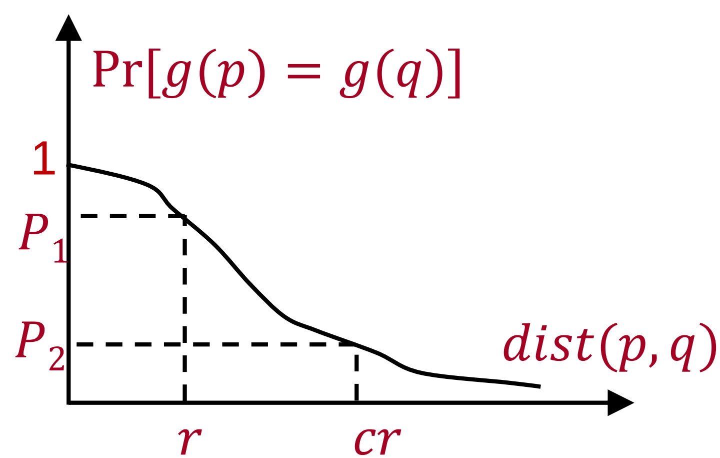
  - *Far* when $\|p - q\| > cr$

$P_2 = \Pr[g(p) = g(q)]$ is "small"



- Use several hash tables: n^ρ, where

$$\rho = \frac{\log 1/P_1}{\log 1/P_2}$$



Locality sensitive hash functions

- Hash function g is usually a concatenation of "primitive" functions:
  - $g(p) = \langle h_1(p), h_2(p), \dots, h_k(p) \rangle$

- Example: Hamming space $\{0,1\}^d$
  - $h(p) = p_j$, i.e., choose j^{th} bit for a random j
  - $g(p)$ chooses k bits at random
  - $\Pr[h(p) = h(q)] = 1 - \frac{Ham(p,q)}{d}$
  - $P_1 = 1 - \frac{r}{d} \approx e^{-r/d}$
  - $P_2 = 1 - \frac{cr}{d} \approx e^{-cr/d}$
  - $\rho = \frac{\log 1/P_1}{\log 1/P_2} = \frac{r/d}{cr/d} = \frac{1}{c}$

Full algorithm

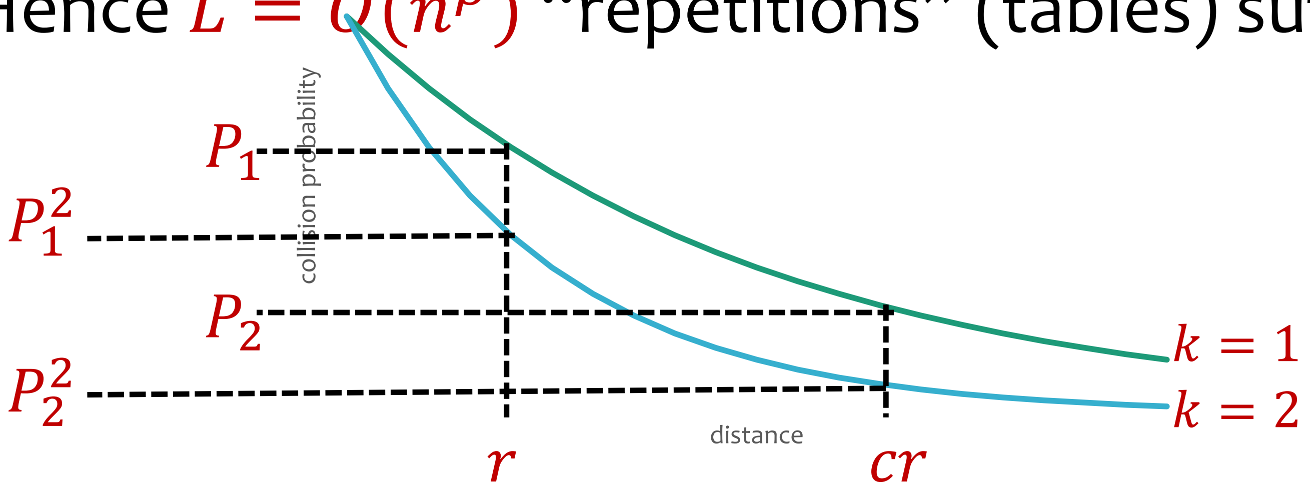
- **Data structure** is just $L = n^\rho$ hash tables:
  - Each hash table uses a fresh random function
    $g_i(p) = \langle h_{i,1}(p), \ldots, h_{i,k}(p) \rangle$
  - Hash all dataset points into the table
- **Query:**
  - Check for collisions in each of the hash tables
  - until we encounter a point within distance cr
- Guarantees:
  - Space: $O(nL) = O(n^{1+\rho})$, plus space to store points
  - Query time: $O(L \cdot (k + d)) = O(n^\rho \cdot d)$ (in expectation)
  - 50% probability of success.

Analysis of LSH Scheme

- Choice of parameters k, L ?
  - L hash tables with $g(p) = \langle h_1(p), \dots, h_k(p) \rangle$

set k s.t.

- Pr[collision of *far* pair] $= P_2^k$ $= 1/n$
- Pr[collision of *close* pair] $= P_1^k = \left(P_2^\rho\right)^k = 1/n^\rho$
- Hence $L = O(n^\rho)$ "repetitions" (tables) suffice!



Analysis: Correctness

- Let p^* be an r-near neighbor
  - If does not exists, algorithm can output anything
- Algorithm fails when:
  - near neighbor p^* is not in the searched buckets $g_1(q), g_2(q), \dots, g_L(q)$
- Probability of failure:
  - Probability q, p^* do not collide in a hash table: $\leq 1 - P_1^k$
  - Probability they do not collide in L hash tables at most

$$\left(1 - P_1^k\right)^L = \left(1 - \frac{1}{n^\rho}\right)^{n^\rho} \leq 1/e$$

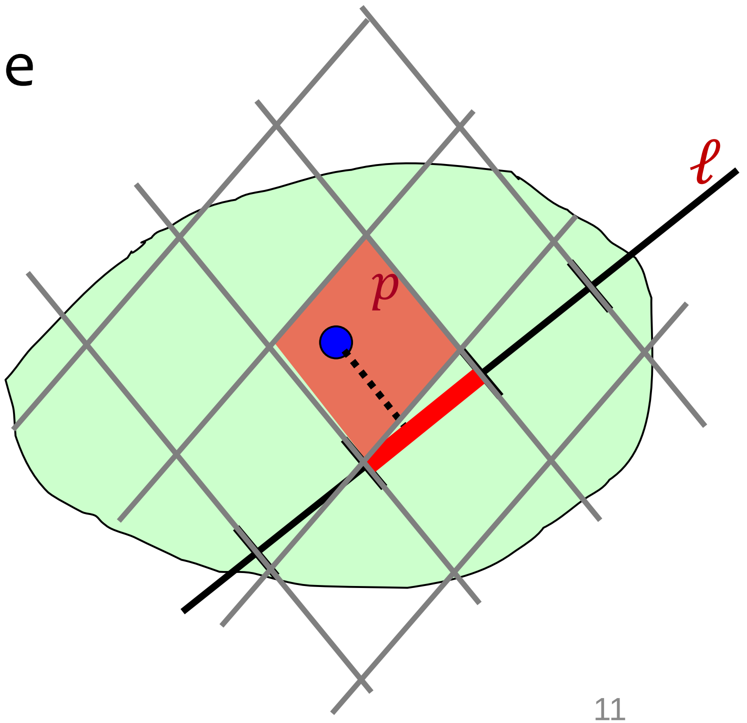
Analysis: Runtime

- Runtime dominated by:
  - Hash function evaluation: $O(L \cdot k)$ time
  - Distance computations to points in buckets
- Distance computations:
  - Care only about far points, at distance $> cR$
  - In one hash table, we have
    - Probability a far point collides is at most $P_2^k = 1/n$
    - Expected number of far points in a bucket: $n \cdot \frac{1}{n} = 1$
  - Over L hash tables, expected number of far points is L
- Total: $O(Lk) + O(Ld) = O(n^\rho(\log n + d))$ in expectation

NNS for Euclidean space

[Datar-Immorlica-Indyk-Mirrokni'04]

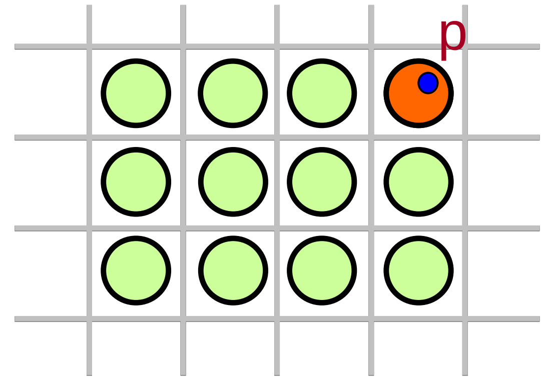
- Hash function g is a concatenation of "primitive" functions:
  - $g(p) = \langle h_1(p), h_2(p), \ldots, h_k(p) \rangle$
- LSH function $h(p)$:
  - pick a random line ℓ, and quantize
  - project point into ℓ
  - $h(p) = \left\lfloor \frac{p \cdot \ell}{w} + b \right\rfloor$
    - ℓ is a random Gaussian vector
    - b random in $[0,1]$
    - w is a parameter (e.g., 4)
- $\rho = 1/c$



Optimal* LSH

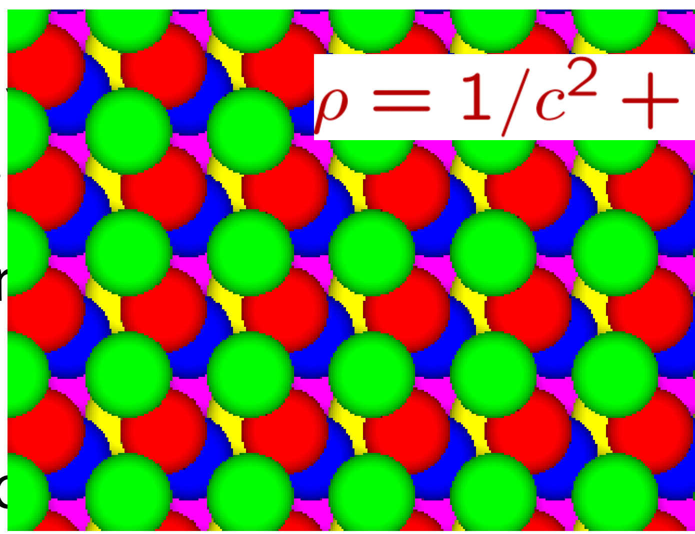
[A-Indyk'06]

- Regular grid → grid of balls
  - p can hit empty space, so take more such grids until p is in a ball

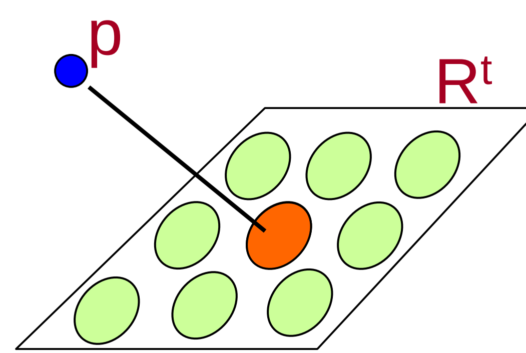


- Need (too) many grids of balls
  - Start by projecting in dimension t

- Anal
- 2D Choi … on t?
  - Tr
  - To



$$\rho = 1/c^2 + o_t(1)$$

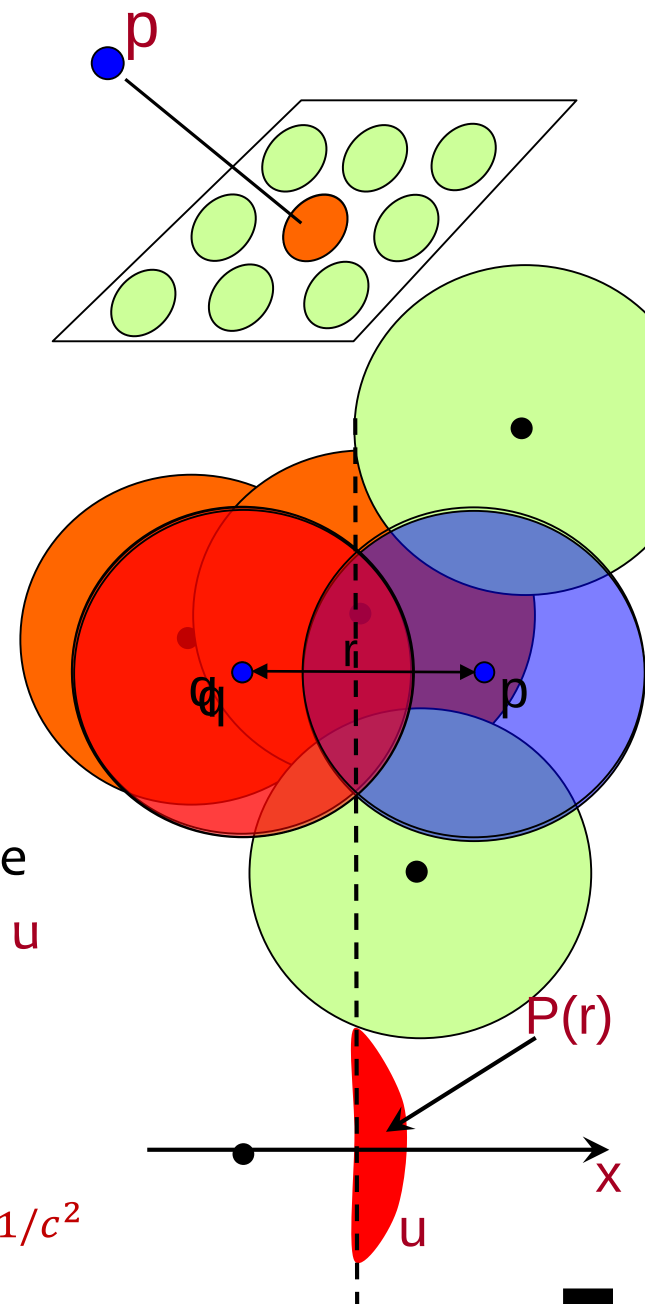


Proof idea

- Claim: $\rho \approx 1/c^2$, i.e.,

$$P(r) \geq P(cr)^{1/c^2}$$

  - P(r)=probability of collision when ||p-q||=r
- Intuitive proof:
  - Projection approx preserves distances [JL]
  - P(r) = intersection / union
  - P(r)≈random point u beyond the dashed line
  - Fact (high dimensions): the x-coordinate of u has a nearly Gaussian distribution
    $\rightarrow$ P(r) ≈ exp(-A·r²)



$$P(r) = \exp(-Ar^2) = [\exp(-A(cr)^2]^{1/c^2} = P(cr)^{1/c^2}$$

LSH Zoo

- Hamming distance
  - h: pick a random coordinate(s) [IM98]
- Manhattan distance:
  - h: random grid [AI'06]
- Jaccard distance between sets:
  - $J(A,B) = \frac{A \cap B}{A \cup B}$
  - h: pick a random permutation π on the universe

    $$h(A) = \min_{a \in A} \pi(a)$$

    min-wise hashing [Bro'97,BGMZ'97]
- Angle: sim-hash [Cha'02,…]

To be or not to be

To Simons or not to Simons

be not or Simons to

…1**1**101…

be not or Simons to

…0**1**111…

…21102…

…01122…

{be,not,or,to}

{not,or,to, Simons}

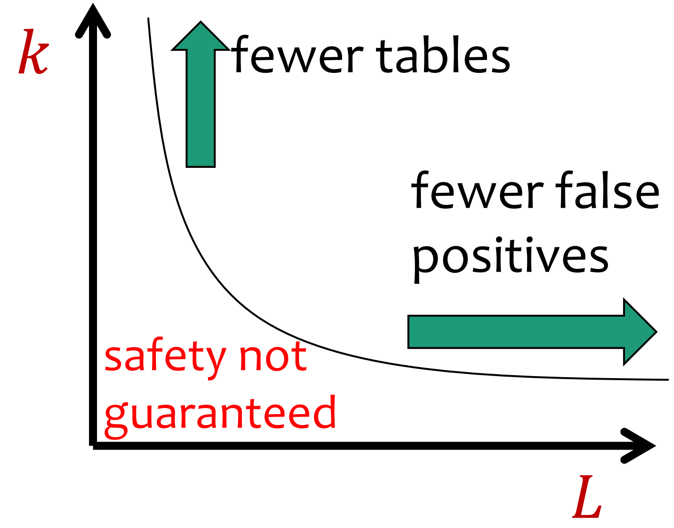
↓

be

↓

to

π=be,to,Simons,or,not



LSH in the wild

- If want exact NNS, what is c?
  - Can choose any parameters L, k
  - Correct as long as $\left(1 - P_1^k\right)^L \leq 0.1$
  - Performance:
    - trade-off between # tables and false positives
    - will depend on dataset "quality"
    - Can tune L, k to optimize for given dataset

Time-Space Trade-offs

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]
				$\rho = 1/c^2$	[DIIM'04, 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}	$\cancel{O(d \log n)}$ 1 mem lookup	$c = 1 + \epsilon$	[KOR'98, IM'98, Pan'06]
		$n^{o(1/\epsilon^2)}$	ω(1) memory lookups		[AIP'06]

LSH is tight…

leave the rest to cell-probe lower bounds?

Data-dependent Hashing!

[A-Indyk-Nguyen-Razenshteyn'14]

- NNS in Hamming space (ℓ_1) with $n^\rho \cdot d$ query time, $n^\rho + nd$ space and preprocessing for
  - $\rho \approx \frac{7/8}{c} + O\left(\frac{1}{c^{3/2}}\right) + o(1)$
  - optimal LSH: $\rho = \frac{1}{c}$ of [IM'98]
- NNS in Euclidean space (ℓ_2) with:
  - $\rho \approx \frac{7/8}{c^2} + O\left(\frac{1}{c^3}\right) + o(1)$
  - optimal LSH: $\rho \approx \frac{1}{c^2}$ of [AI'06]

A look at LSH lower bounds

- LSH lower bounds in Hamming space
  - Fourier analytic [O'Donnell-Wu-Zhou'11]
- ~~[Motwani-Naor-Panigrahy'06]~~
  - H distribution over hash functions $h: \{0,1\}^d \to U$
  - Far pair: p, q random, distance $= \cancel{d/2}$ ϵd
  - Close pair: p, q random at distance = $\cancel{\frac{d/2}{c}}$ $\frac{\epsilon d}{c}$
  - Get $\cancel{\rho \geq 0.5/c}$
    $\rho \geq 1/c$

Why not NNS lower bound?

- Suppose we try to generalize [OWZ'11] to NNS
  - Pick random q
  - All the "far point" are concentrated in a small ball of radius ϵd
  - Easy to see at preprocessing: actual near neighbor close to the center of the minimum enclosing ball

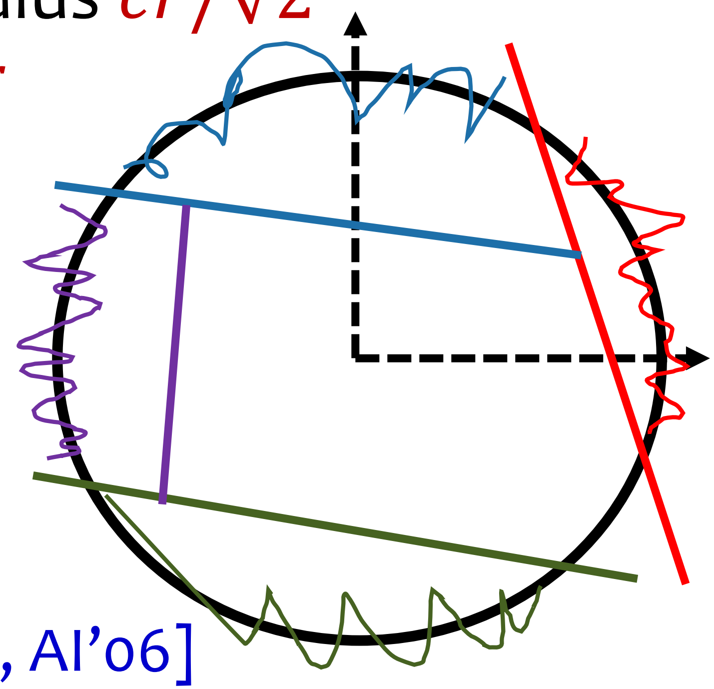
Intuition

- Data dependent LSH:
  - Hashing that *depends on the entire given dataset*!

- Two components:
  - "Nice" geometric configuration with $\rho < 1/c^2$
  - Reduction from general to this "nice" geometric configuration

Nice Configuration: “sparsity”

- All points are on a sphere of radius $cr/\sqrt{2}$
  - Random points are at distance cr



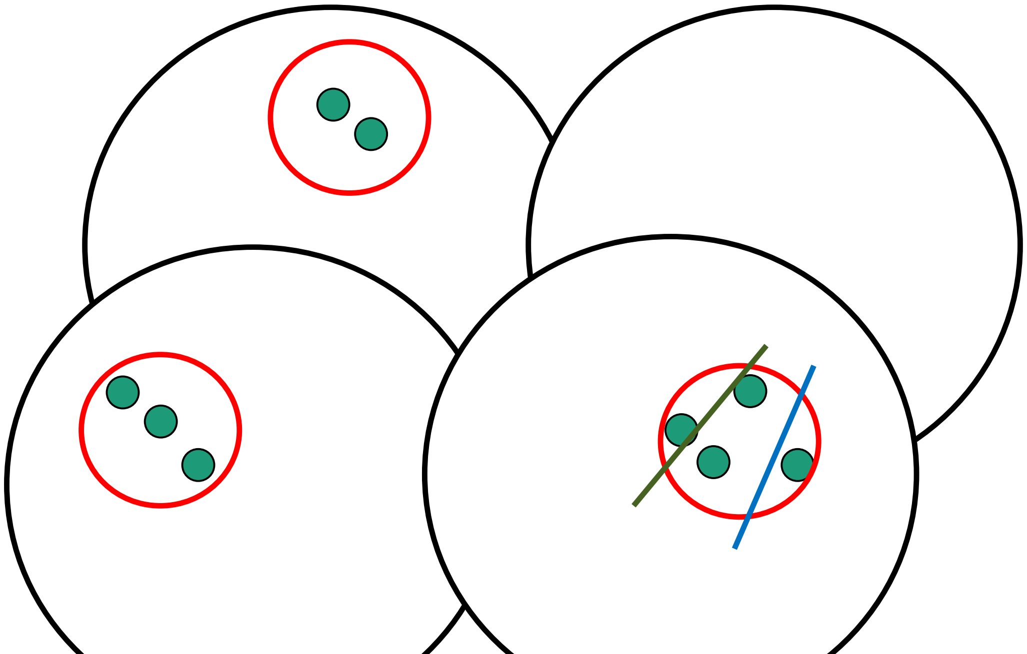
- Lemma 1: $\rho \approx \frac{0.5}{c^2}$
- “Proof”:
  - Obtained via “cap carving”
  - Similar to “ball carving” [KMS’98, AI’06]
- Lemma 1’: $\rho \approx \left(1 - \frac{1}{4\eta^2}\right)\frac{1}{c^2}$ for radius = ηcr

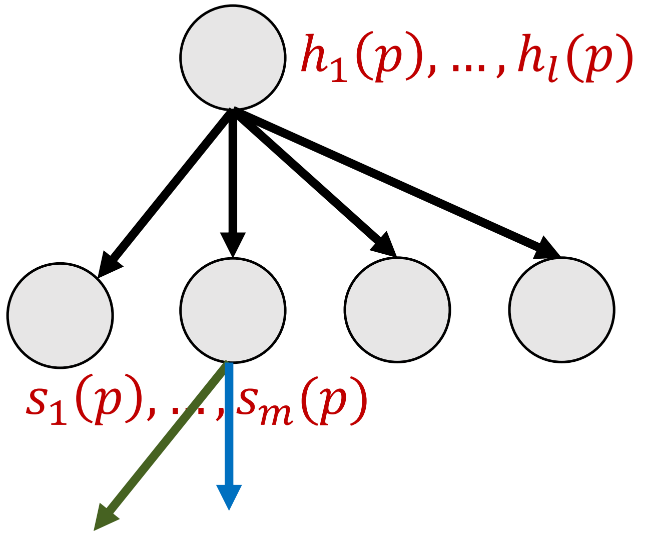
Reduction: into spherical LSH

- Idea: apply a few rounds of "regular" LSH
  - Ball carving [AI'06]
  - as if target approx. is 5c => query time $n^{1/25c^2}$
- Intuitively:
  - far points unlikely to collide
  - partitions the data into buckets of small diameter $\approx 5cr$
  - find the minimum enclosing ball
  - finally apply spherical LSH on this ball!

Two-level algorithm

- n^ρ hash tables, each with:
  - hash function $g = (h_1, h_2, \dots h_l, s_1, \dots s_m)$
  - h_i's are "ball carving LSH" (data *independent*)
  - s_j's are "spherical LSH" (data *dependent*)
- Final ρ is an "average" of ρ from levels 1 and 2





Details

- Inside a bucket, need to ensure "sparse" case
  - 1) drop all "far pairs"
  - 2) find minimum enclosing ball (MEB)
  - 3) partition by "sparsity" (distance from center)

1) Far points

- In level 1 bucket:
  - Set parameters as if looking for approximation $\tau c/\sqrt{2}$
  - Probability a pair survives:
  - At distance τc: n^{-2}
  - At distance c: n^{-2/τ^2}
  - At distance 1: $n^{-2/\tau^2 c^2}$

  - Expected number of collisions for distance τc is constant
    - Throw out all pairs that violate this

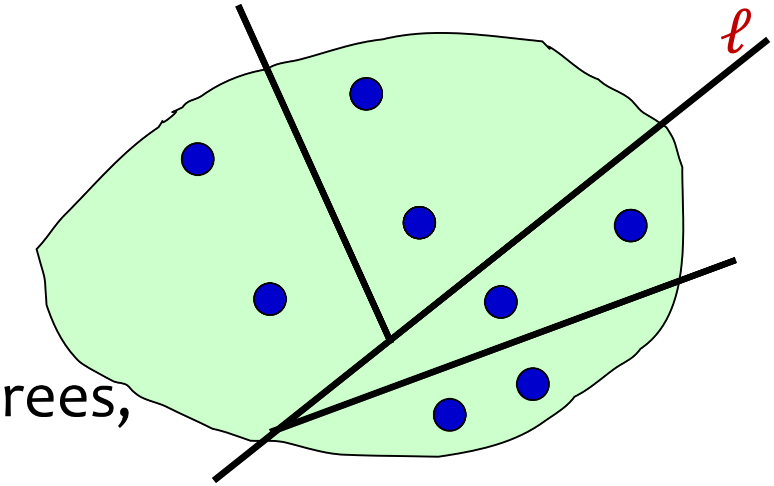
2) Minimum Enclosing Ball

- Use *Jung theorem*:
    - A pointset with diameter τc has a MEB of radius $\tau c/\sqrt{2}$

3) Partition by “sparsity”

- Inside a bucket, points are at distance **at most** $R = \tau c$
- “Sparse” LSH does not work in this case
  - Need to partition by the distance to center
  - Partition into spherical shells of width 1

Practice of NNS



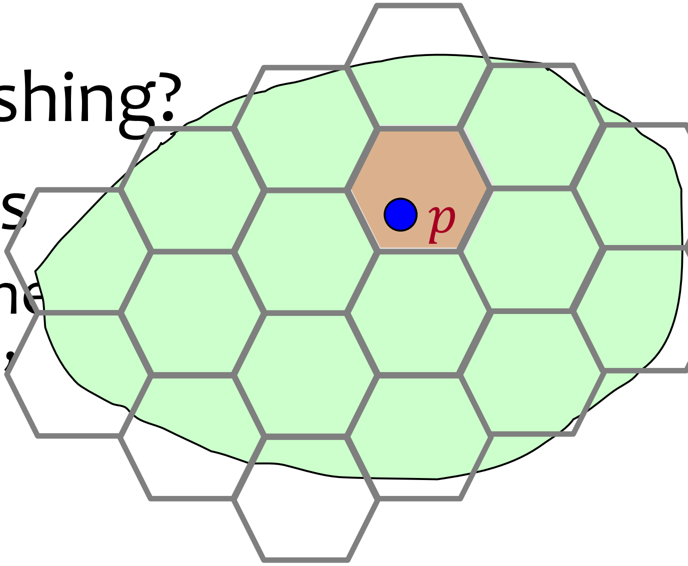
- *Data-dependent* partitions…
- Practice:
  - Trees: kd-trees, quad-trees, ball-trees, rp-trees, PCA-trees, sp-trees…
  - often *no guarantees*
- Theory?
  - assuming more about data: PCA-like algorithms "work" [Abdullah-A-Kannan-Krauthgamer'14]

Finale

- LSH: geometric hashing
- Data-dependent hashing can do better!
  - Better upper bound?
    - Multi-level improves a bit, but not too much
    - $\rho = \frac{0.5}{c^2}$ for ℓ_2 ?
- Best partitions in theory and practice?
- LSH for other metrics:
  - Edit distance, Earth-Mover Distance, etc
- Lower bounds (cell probe?)

Open question:

- Practical variant of ball-carving hashing?
- Design space-partition of $\mathfrak{R}^t$ that is
  - efficient: point location in poly(t) time
  - qualitative: regions are “sphere-like”



[Prob. needle of length 1 is not cut]

$\geq$

[Prob needle of length c is not cut]$^{1/c^2}$