Sketching and Nearest Neighbor Search (2)

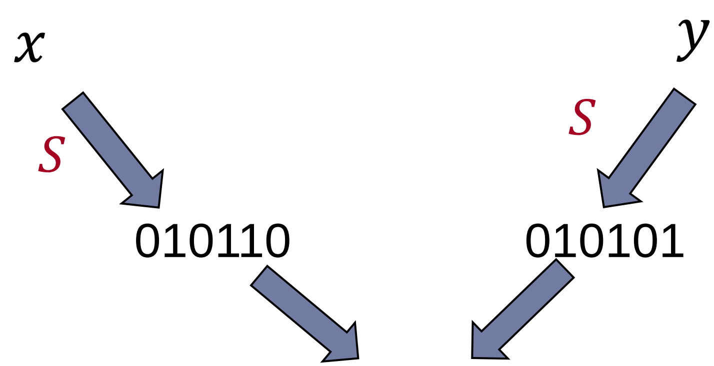
Alex Andoni

(Columbia University)

MADALGO Summer School on Streaming Algorithms 2015

Sketching

- $S: \Re^d \to$ short bit-strings
  - given $S(x)$ and $S(y)$, should be able to estimate some function of x *and* y
- ℓ_2, ℓ_1 norm: $O(\epsilon^{-2})$ *words*
- Decision version: given r in advance…
  - ℓ_2, ℓ_1 norm: $O(\epsilon^{-2})$ *bits*



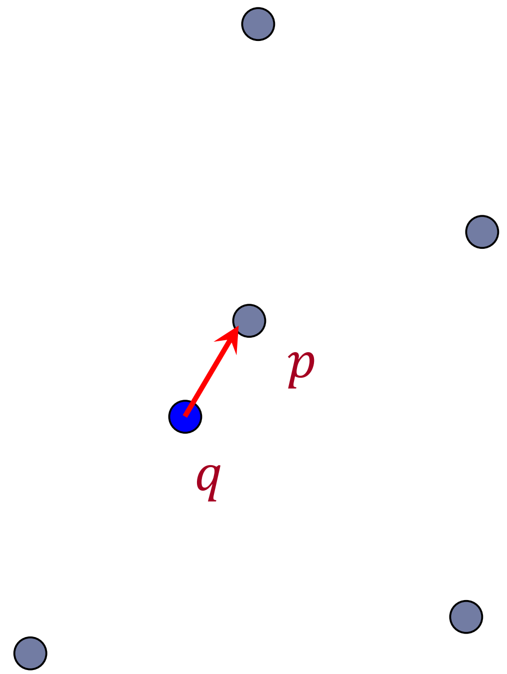
Distinguish between
$||x - y|| \le r$
$||x - y|| > (1 + \epsilon)r$

Sketching: decision version

- Consider Hamming space: $x, y \in \{0,1\}^d$
- Lemma: for any r, can achieve $O(1/\epsilon^2)$ bit sketch.

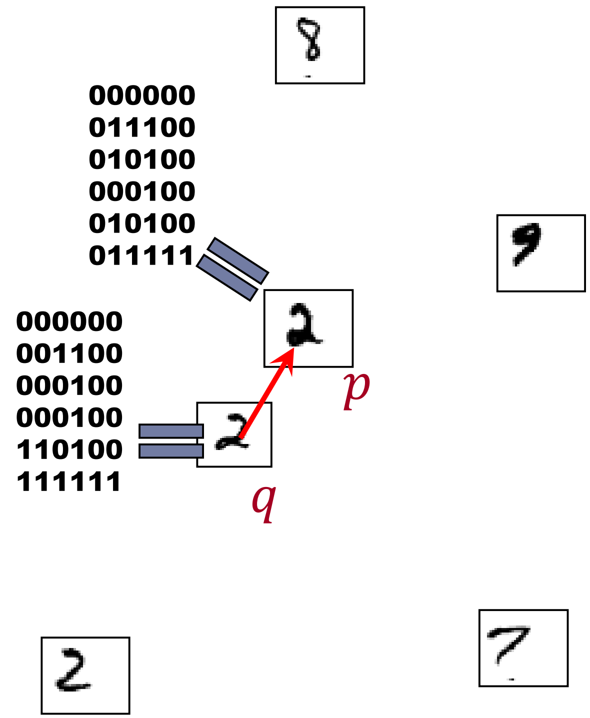
Nearest Neighbor Search (NNS)

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



Motivation

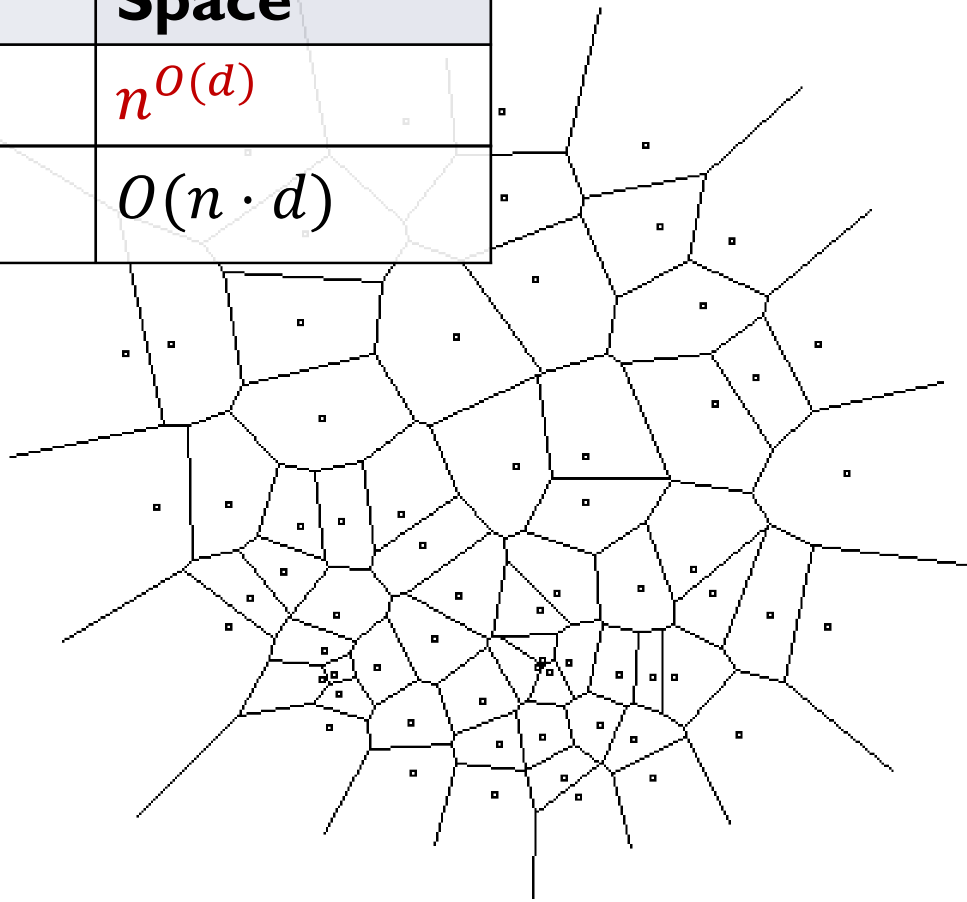
- Generic setup:
  - Points model *objects (e.g. images)*
  - Distance models *(dis)similarity measure*
- Application areas:
  - machine learning: k-NN rule
  - image/video/music recognition, deduplication, bioinformatics, etc…
- Distance can be:
  - Hamming, Euclidean, …
- Primitive for other problems:
  - find the similar pairs, clustering…



Curse of Dimensionality

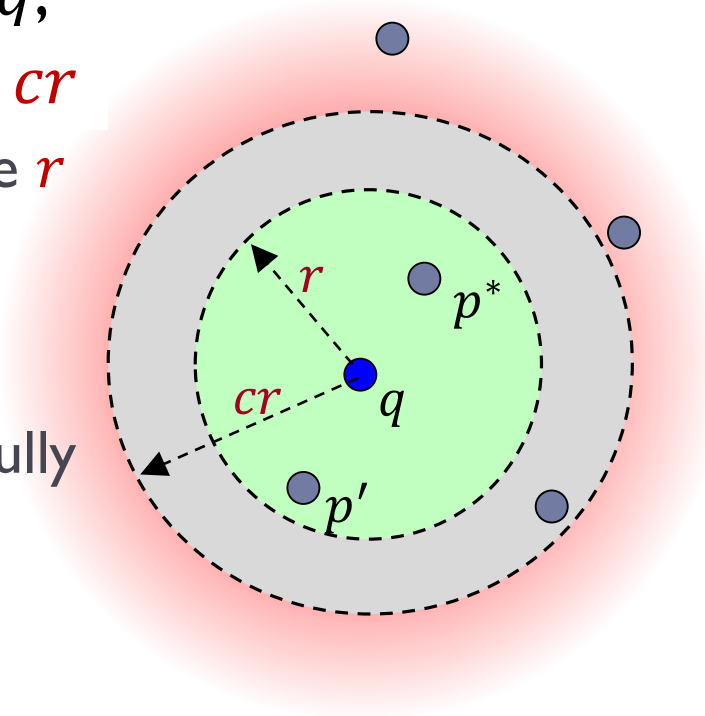
- All exact algorithms degrade rapidly with the dimension d

Algorithm	Query time	Space
Voronoi diagram	$(d \cdot \log n)^{O(1)}$	$n^{O(d)}$
Linear scan	$O(n \cdot d)$	$O(n \cdot d)$



Approximate NNS

- c-approximate r-near neighbor: given a query point q, report a point $p' \in P$ s.t. $||p' - q|| \le cr$
  - assuming there is a point within distance r
- Practice: use for exact NNS
  - *Filtering*: gives a set of candidates (hopefully small)



NNS algorithms

Dependence on dimension:

- Exponential

  [Arya-Mount'93], [Clarkson'94], [Arya-Mount-Netanyahu-Silverman-We'98], [Kleinberg'97], [Har-Peled'02],[Arya-Fonseca-Mount'11],…

- Linear/polynomial

  [Kushilevitz-Ostrovsky-Rabani'98], [Indyk-Motwani'98], [Indyk'98, '01], [Gionis-Indyk-Motwani'99], [Charikar'02], [Datar-Immorlica-Indyk-Mirrokni'04], [Chakrabarti-Regev'04], [Panigrahy'06], [Ailon-Chazelle'06], [A.-Indyk'06], [A.-Indyk-Nguyen-Razenshteyn'14], [A.-Razenshteyn'15]

NNS via sketching: Approach 1

- Boosted sketch:
  - Let S = sketch for the decision version (90% success prob)
  - new sketch $W : k = O(\log n)$ copies of S

    estimator is the median of the k estimators
  - Sketch size: $O(\epsilon^{-2} \log n)$
  - Success probability: $1 - n^{-2}$
- Preprocess: compute sketches $W(p)$ for all the points $p \in D$
- Query: compute sketch $W(q)$, and compute distance to all points using sketch

- Time: improved from $O(nd)$ to $O(n\epsilon^{-2} \log n)$

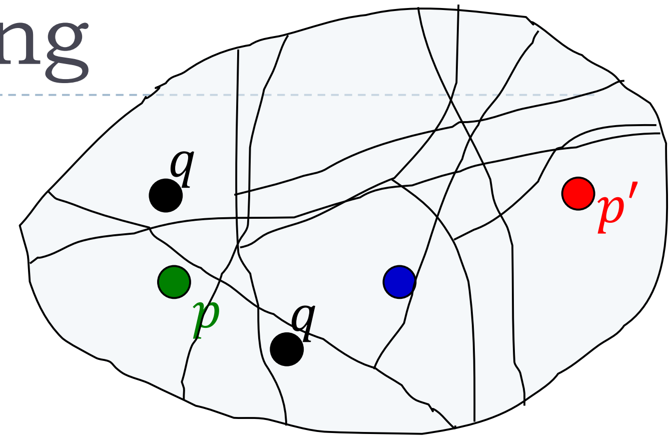
NNS via sketching: Approach 2

- Query time below n ?
- Theorem [KOR98]: $O(d\epsilon^{-2}\log n)$ query time and $n^{O(1/\epsilon^2)}$ space for $1+\epsilon$ approximation.
- Proof:
  - Note that $W(q)$ has $w = O(\epsilon^{-2}\log n)$ bits
  - Only 2^w possible sketches!
  - Store an answer for each of $2^w = n^{O(\epsilon^{-2})}$ possible inputs
- If a distance has constant-size sketch, admits a poly-space NNS data structure!

- Space closer to linear?
  - approach 3 will require more specialized sketches…

3: Locality Sensitive Hashing

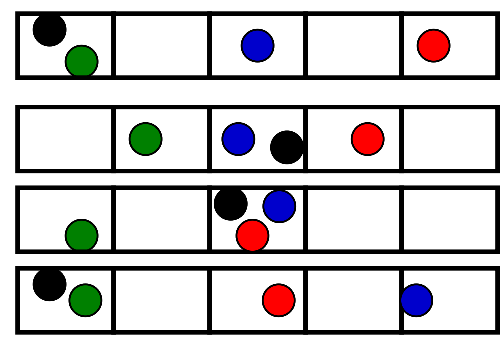
[Indyk-Motwani'98]



Random hash function h on R^d satisfying:

- for *close pair* (when $\|q-p\| \le r$)

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

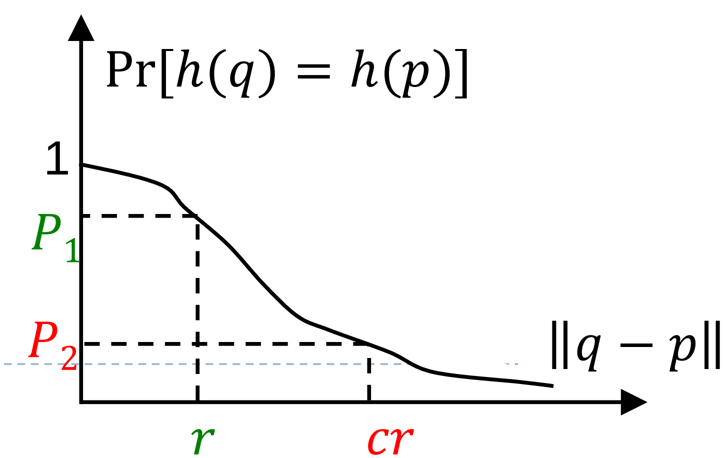


- for *far pair* (when $\|q-p'\| > cr$)

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

Use several hash tables

$$n^\rho, \text{ where } \rho = \frac{\log 1/P_1}{\log 1/P_2}$$



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

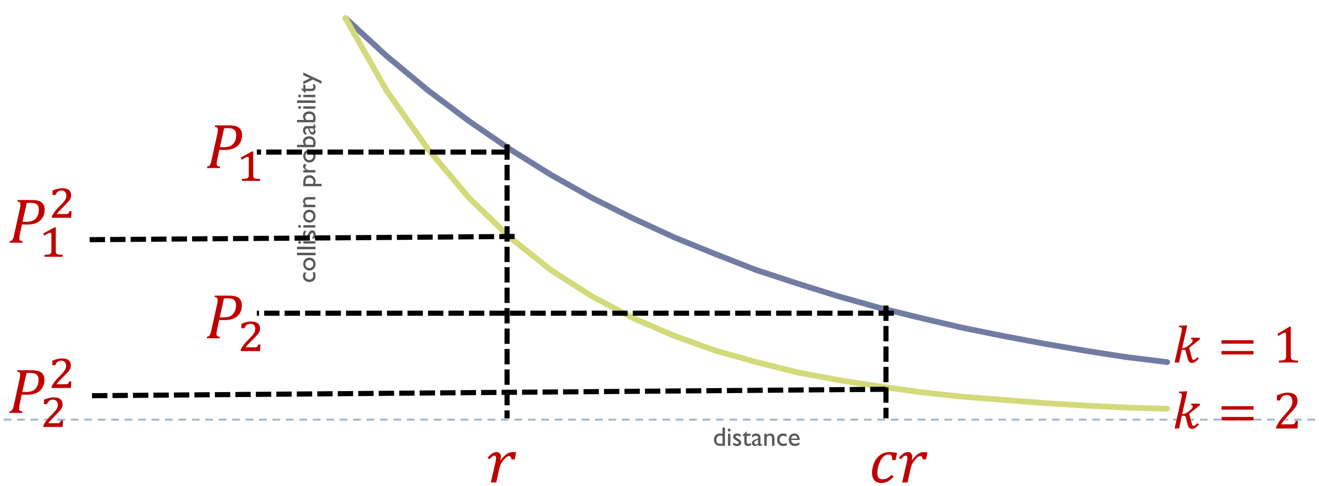
- **Data structure** is just $L = n^\rho$ hash tables:
  - Each hash table uses a fresh random function
    $g_i(p) = \langle h_{i,1}(p), \dots, h_{i,k}(p) \rangle$
  - Hash all dataset points into the table
- **Query**:
  - Check for collisions in each of the hash tables
  - until we encounter a point 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 $g(p) = \langle h_1(p), \ldots, h_k(p) \rangle$

set k s.t.
= $1/n$

- 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 $g_1(q), g_2(q), \ldots, g_L(q)$
- 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

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

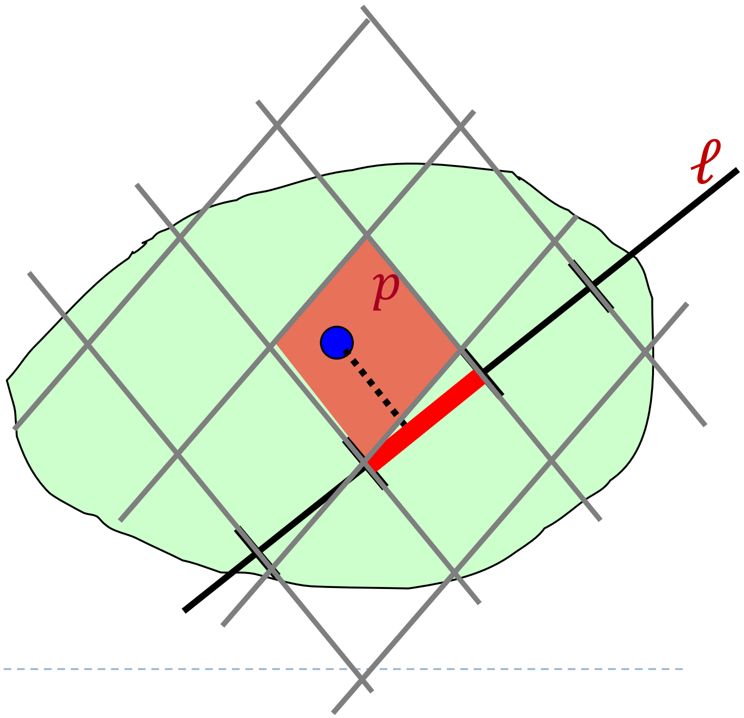
Analysis: Runtime

- 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]

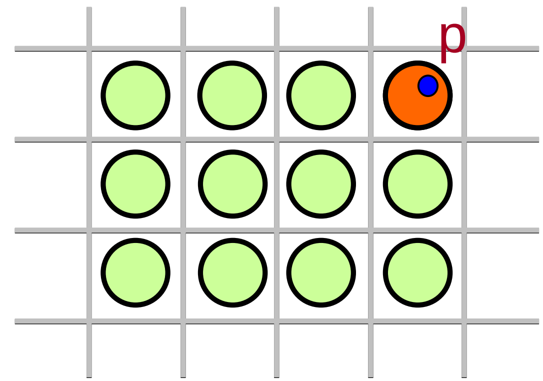
- Hash function g is a concatenation of "primitive" functions:
  - $g(p) = \langle h_1(p), h_2(p), \ldots, h_k(p) \rangle$
- LSH function $h(p)$:
  - pick a random line ℓ, and quantize
  - project point into ℓ
  - $h(p) = \left\lfloor \frac{p \cdot \ell}{w} + b \right\rfloor$
    - ℓ is a random Gaussian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



Optimal Euclidean LSH

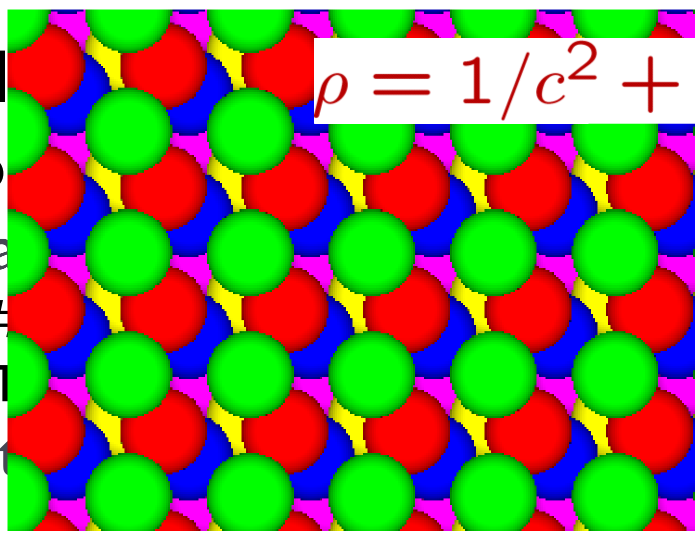
[A-Indyk'06]

- 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

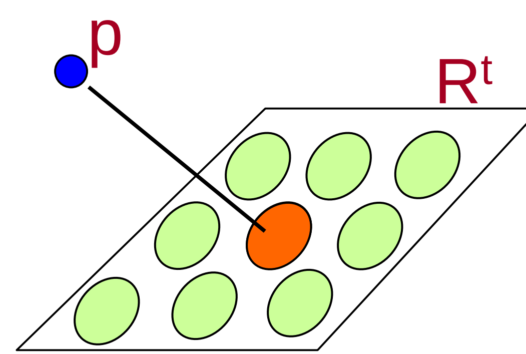


- Anal
- Cho … on t?
  - Tra
    - #
    - T
  - Tot

2D



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



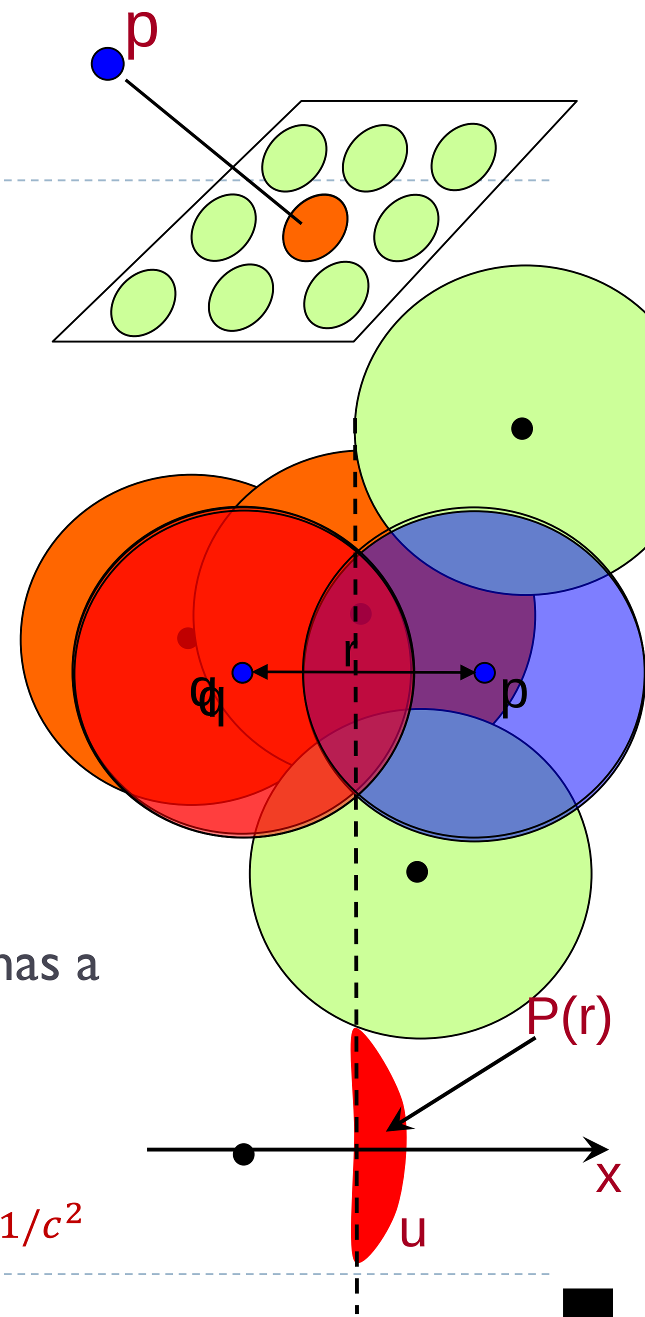
Proof idea

- **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

    $\rightarrow$ $P(r) \approx \exp(-A \cdot r^2)$

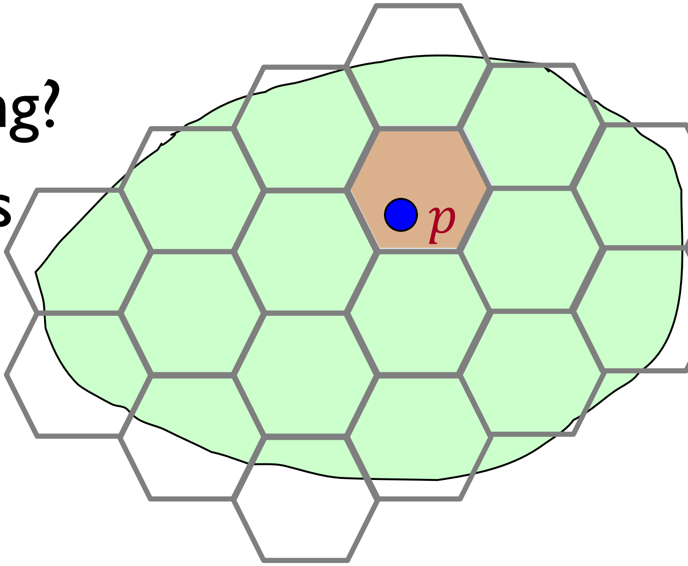


$$P(r) = \exp(-Ar^2) = [\exp(-A(cr)^2]^{1/c^2} = P(cr)^{1/c^2}$$

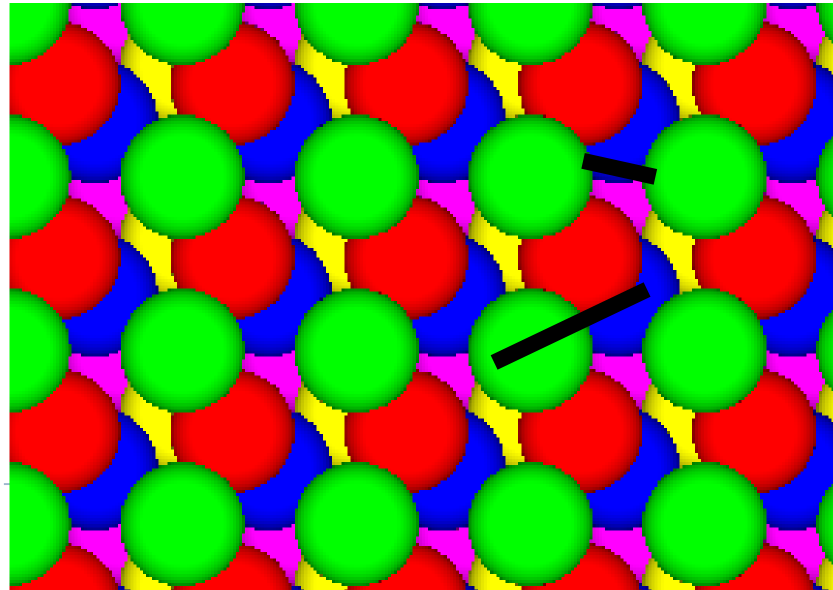
■

Open question:

- More practical variant of above hashing?
- Design space partitioning of $\Re^t$ that is
  - efficient: point location in poly(t) time
  - qualitative: regions are "sphere-like"



$$[\text{Prob. needle of length } 1 \text{ is not cut}]^{c^2} \geq [\text{Prob needle of length } c \text{ is not cut}]$$



LSH Zoo

- Hamming distance [IM'98]
  - h: pick a random coordinate(s)
- Manhattan distance [AI'06]
  - h: cell in a randomly shifted grid
- 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]

…

To be or not to be

To sketch or not to sketch

be not or sketch to

…11101…

…01111…

…21102…

…01122…

{be,not,or,to}

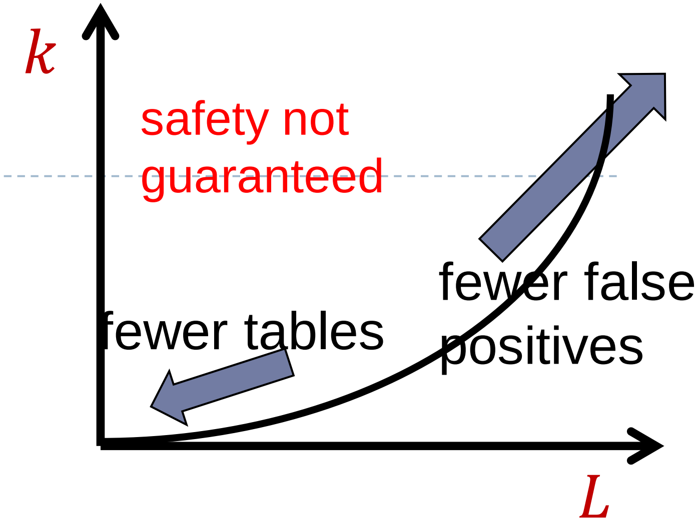
{not,or,to, sketch}

be

to

π=be,to,sketch,or,not

LSH in practice



- 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"