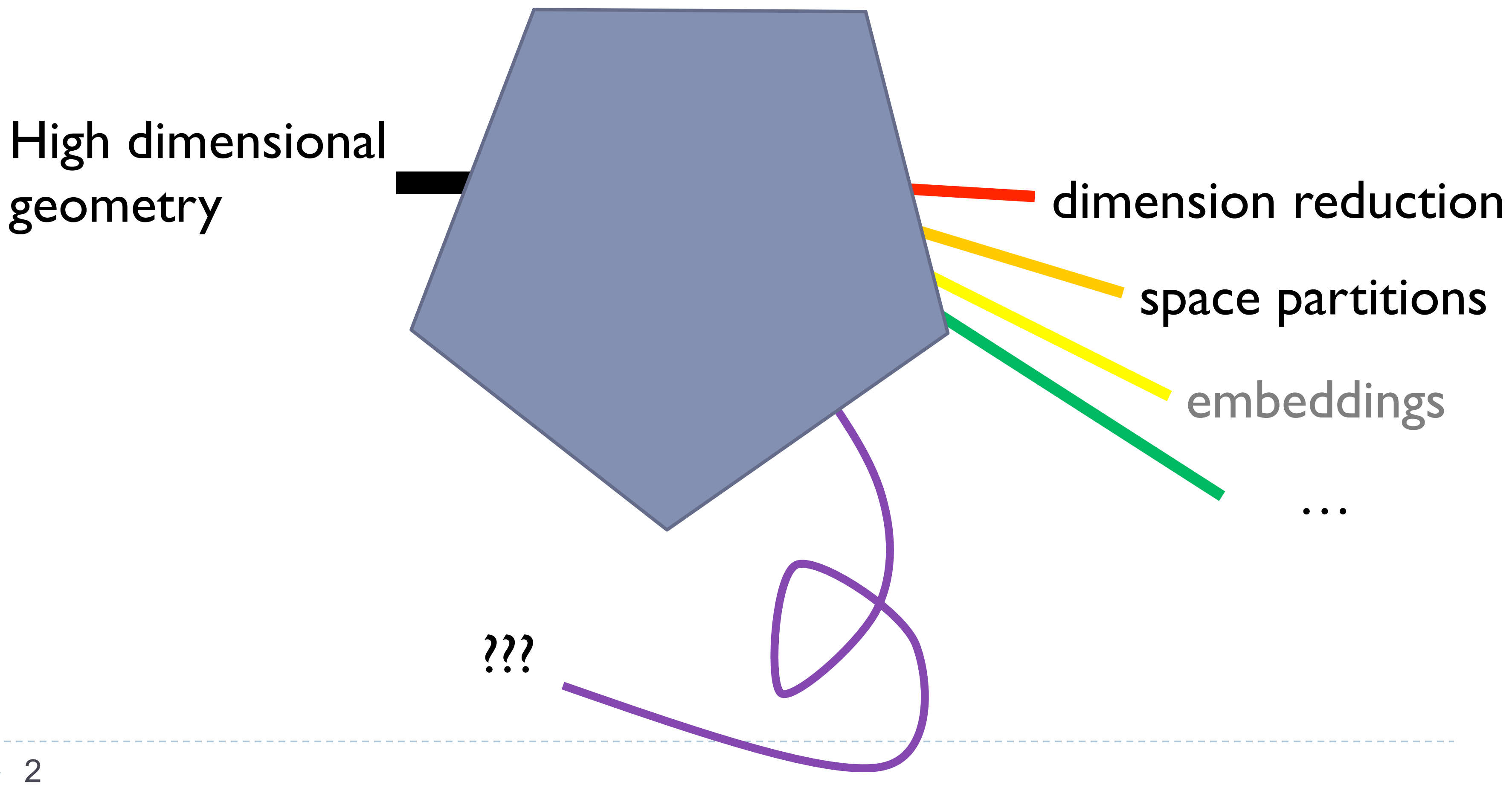
Algorithmic High-Dimensional Geometry 1

Alex Andoni

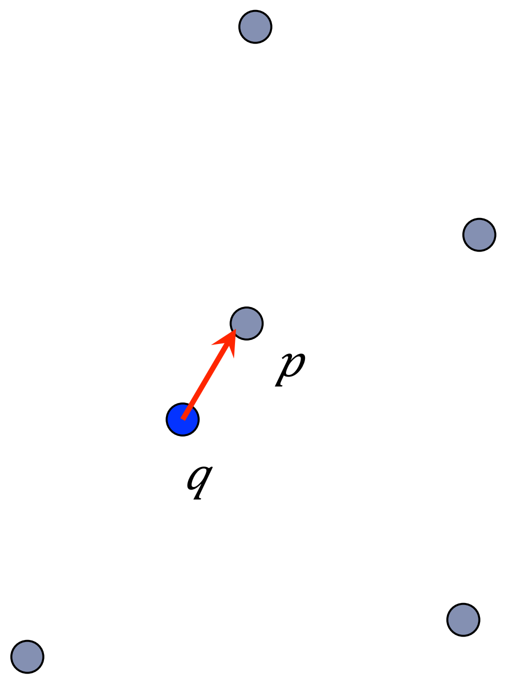
(Microsoft Research SVC)

Prism of nearest neighbor search (NNS)



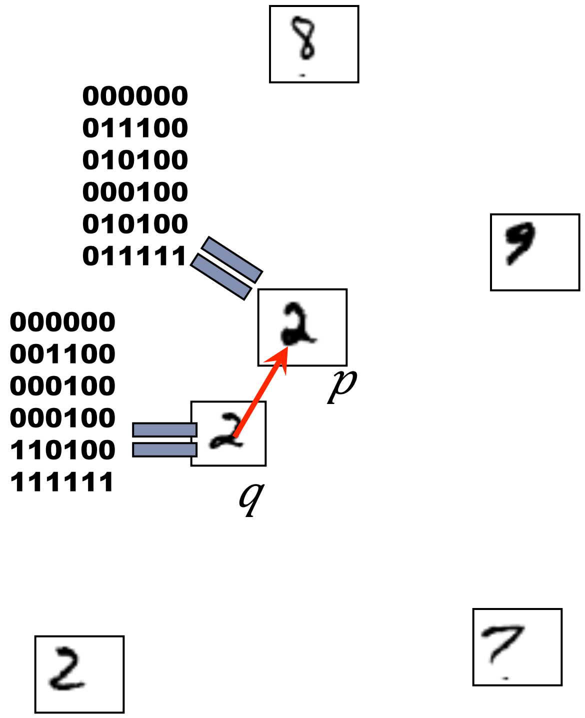
Nearest Neighbor Search (NNS)

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



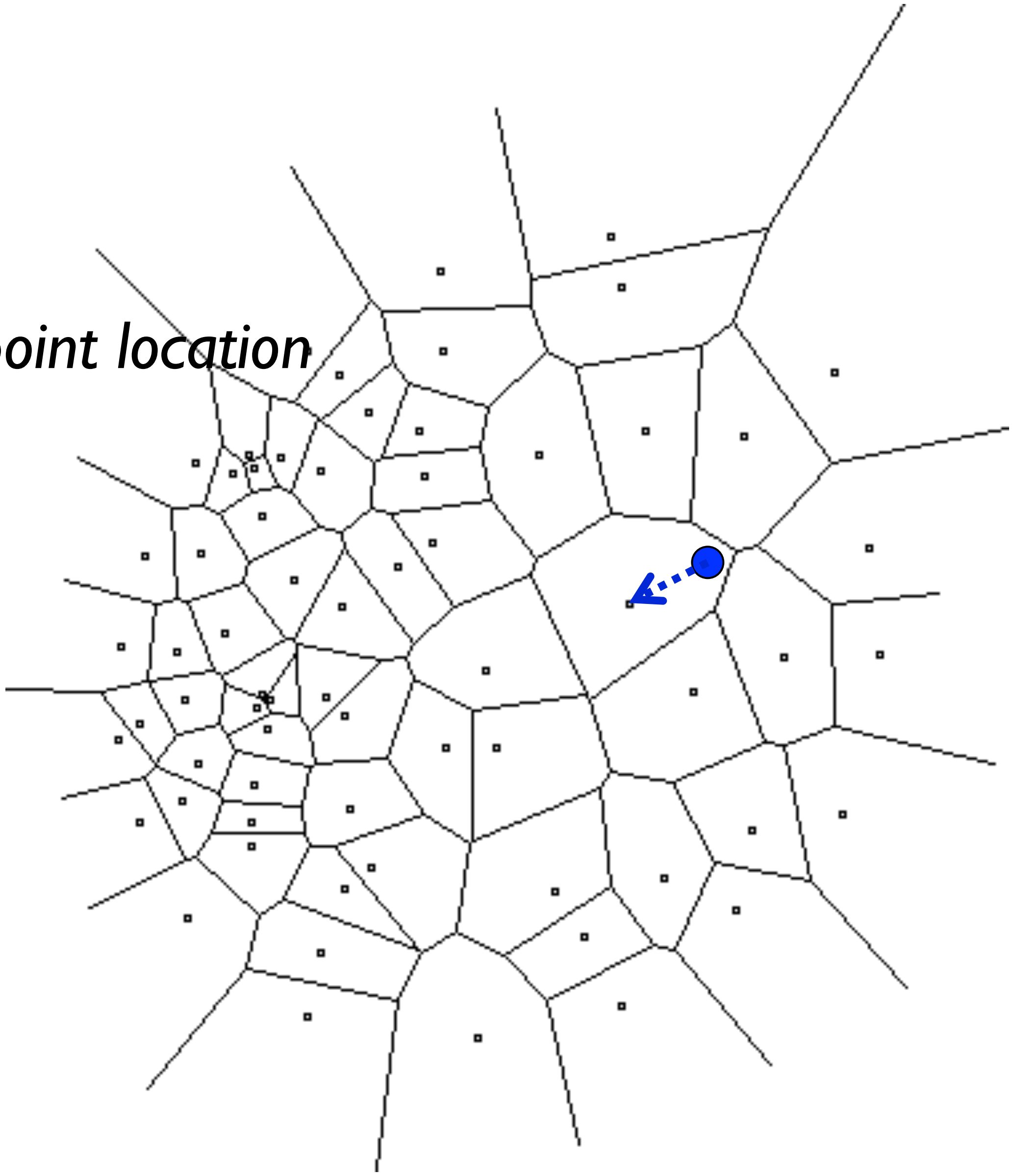
Motivation

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



2D case

- Compute *Voronoi diagram*
- Given query q, perform *point location*
- Performance:
  - Space: $O(n)$
  - Query time: $O(\log n)$



High-dimensional case

- All exact algorithms degrade rapidly with the dimension d

Algorithm	*Query time*	*Space*
Full indexing	$O(\log n \cdot d)$	$n^{O(d)}$ (Voronoi diagram size)
No indexing – linear scan	$O(n \cdot d)$	$O(n \cdot d)$

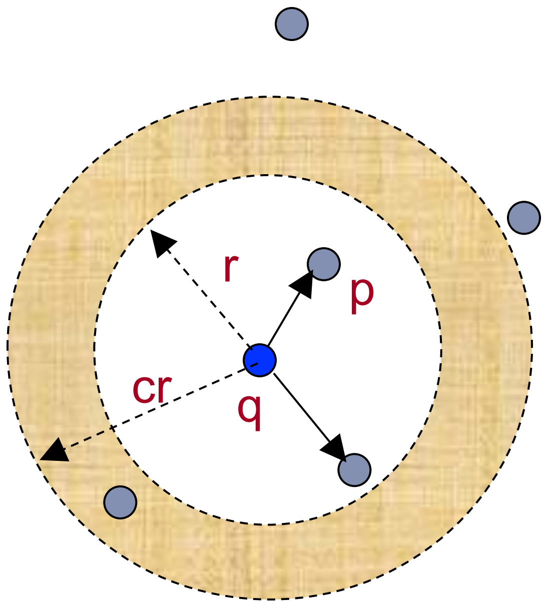
Approximate NNS

c-approximate

- r-near neighbor: given a new point q, report a point p [illegible] D s.t. $||p-q|| \leq r$ cr

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

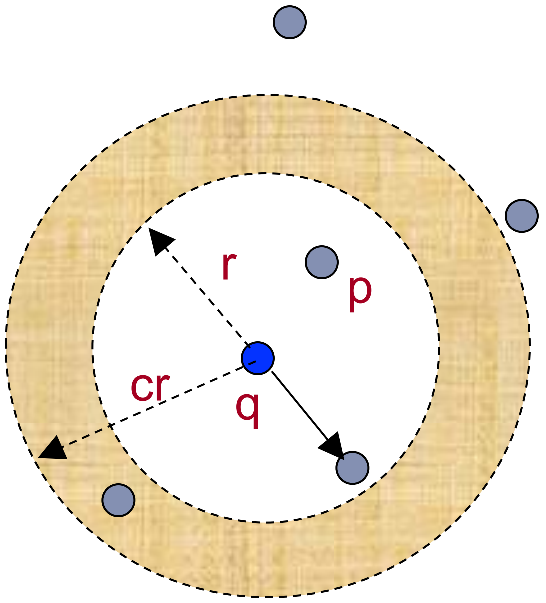
- Randomized: a point p returned with 90% probability



Heuristic for Exact NNS

c-approximate

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



Approximation Algorithms for NNS

- A vast literature:
  - milder dependence on dimension

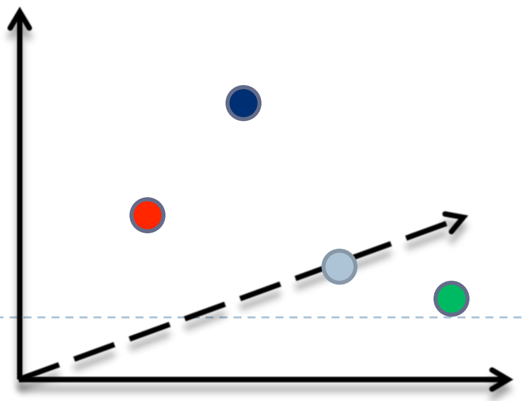
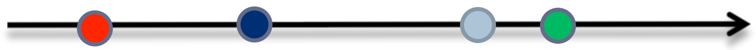
    [Arya-Mount'93], [Clarkson'94], [Arya-Mount-Netanyahu-Silverman-We'98], [Kleinberg'97], [Har-Peled'02], [Chan'02]…
  - little to no dependence on dimension

    [Indyk-Motwani'98], [Kushilevitz-Ostrovsky-Rabani'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], …

Dimension Reduction

Motivation

- If high dimension is an issue, reduce it?!
  - "flatten" dimension d into dimension $k \ll d$
- Not possible in general: packing bound
- But can if: for a fixed subset of $\Re^d$
  - Johnson Lindenstrauss Lemma [JL'84]
- Application: NNS in $\Re^d$
  - Trivial scan: $O(n \cdot d)$ query time
  - Reduce to $O(n \cdot k) + T_{dim-red}$ time if preprocess, where $T_{dim-red}$ time to reduce dimension of the query point

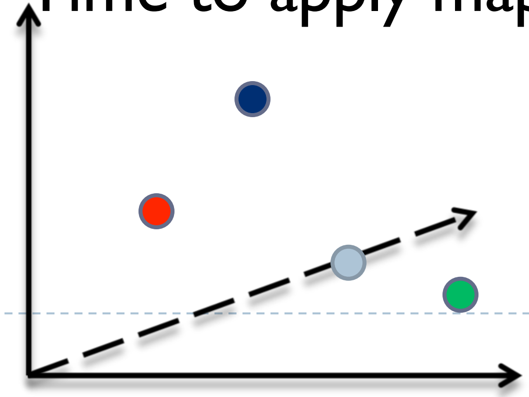



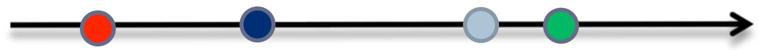
Dimension Reduction

- Johnson Lindenstrauss Lemma: there is a randomized linear map $F: \ell_2^d \to \ell_2^k$, $k \ll d$, that preserves distance between two vectors x, y
  - up to $1+\epsilon$ factor:

$$||x-y|| \le ||F(x)-F(y)|| \le (1+\epsilon)\cdot||x-y||$$

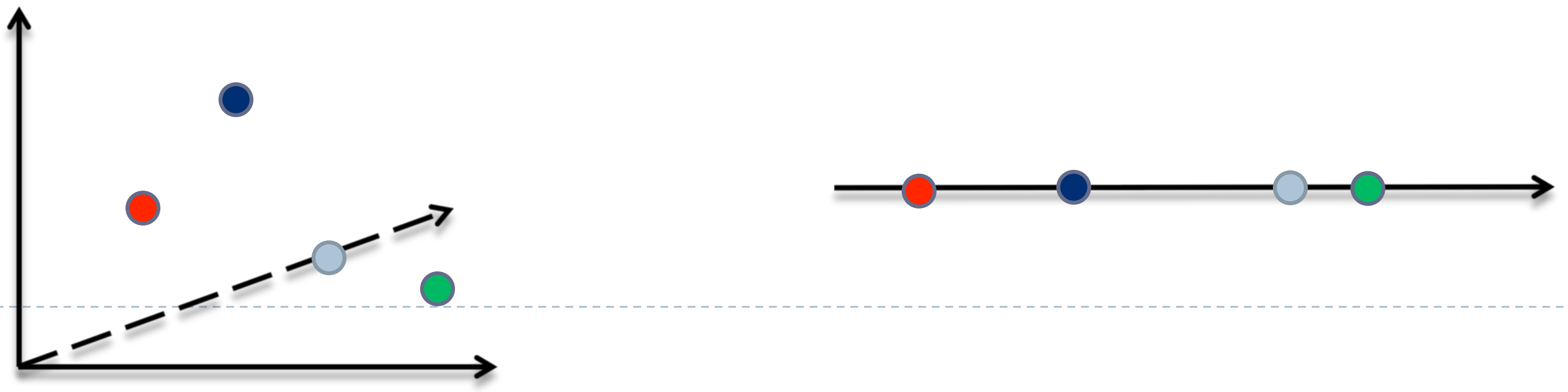
  - with $1-e^{-C\epsilon^2 k}$ probability (C some constant)
- Preserves distances among n points for $k=O(\log n/\epsilon^2)$
- Time to apply map: $T_{dim-red}=O(kd)$



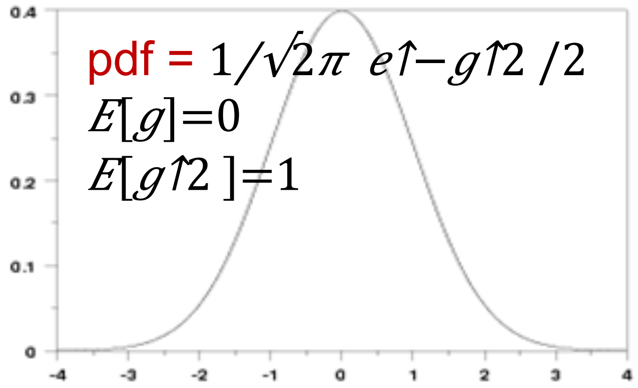


Idea:

- Project onto a *random* subspace of dimension k!



1D embedding



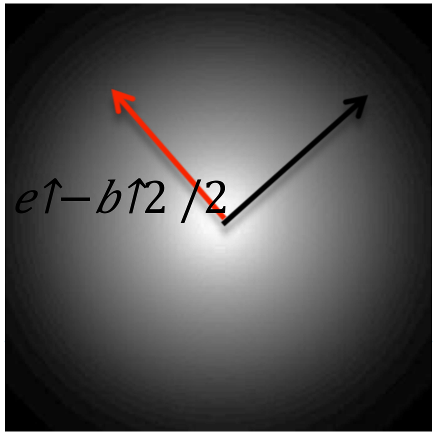
pdf = $\frac{1}{\sqrt{2\pi}} e^{-g^2/2}$

$E[g]=0$

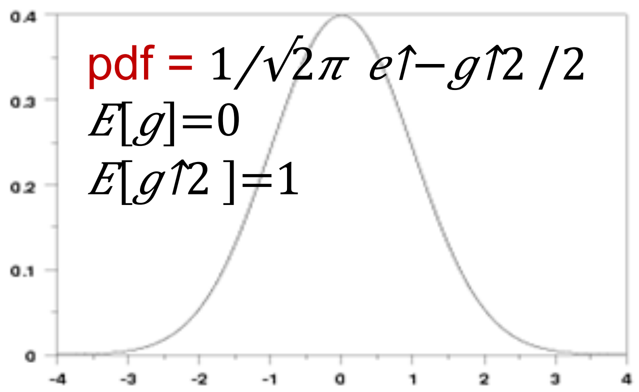
$E[g^2]=1$

- How about one dimension ($k=1$) ?
- Map $f: \ell_2^d \to \mathbb{R}$
  - $f(x)=\sum_i g_i \cdot x_i$,
    - where g_i are iid normal (Gaussian) random variable
- Why Gaussian?
  - Stability property: $\sum_i g_i \cdot x_i$ is distributed as $\|x\| \cdot g$, where g is also Gaussian
  - Equivalently: $\langle g_1, \ldots, g_d \rangle$ is centrally distributed, i.e., has random direction, and projection on random direction depends only on length of x

$$P(a)\cdot P(b)= \\ = \frac{1}{\sqrt{2\pi}} e^{-a^2/2} \frac{1}{\sqrt{2\pi}} e^{-b^2/2} \\ = \frac{1}{2\pi} e^{-(a^2+b^2)/2}$$



1D embedding



pdf = $1/\sqrt{2\pi}\; e^{-g^2/2}$

$E[g]=0$

$E[g^2]=1$

- Map $f(x)=\sum_i g_i \cdot x_i$,
  - for any x, $f(x) \sim \|x\| \cdot g$
  - Linear: $f(x)-f(y)=f(x-y)$
- Want: $|f(x)-f(y)| \approx \|x-y\|$
- Ok to consider $z=x-y$ since f linear
  - $|f(z)|^2 \approx \|z\|^2$
- Claim: for any x,y $\Re^d$, we have
  - Expectation: $[|f(z)|^2] = \|z\|^2$
  - Standard deviation:
    - $[|(f(z)|^2] = O(\|z\|^2)$
- Proof:
  - Expectation = $[(f(z))^2] =$ $[\|z\|^2 \cdot g^2]$ $= \|z\|^2$

Full Dimension Reduction

- Just repeat the 1D embedding for k times!
  - $F(x) = (g_1 \cdot x, g_2 \cdot x, \ldots g_k \cdot x)/\sqrt{k} = 1/\sqrt{k}\ Gx$
  - where G is $k \times d$ matrix of Gaussian random variables

- Again, want to prove:
  - $\|F(z)\| = (1 \pm \epsilon) * \|z\|$
  - for fixed $z = x - y$
  - with probability $1 - e^{-\Omega(\epsilon^2 k)}$

Concentration

- $F(z)$ is distributed as
  - $1/\sqrt{k}\ (||z||\cdot a_1, ||z||\cdot a_2, \ldots ||z||\cdot a_k)$
  - where each a_i is distributed as Gaussian
- Norm $||F(z)||^2 = ||z||^2 \cdot 1/\sqrt{k}\ \sum_i a_i^2$
  - $\sum_i a_i^2$ is called chi-squared distribution with k degrees
- Fact: chi-squared very well concentrated:
  - Equal to $1+\epsilon$ with probability $1-e^{-\Omega(\epsilon^2 k)}$
  - Akin to central limit theorem

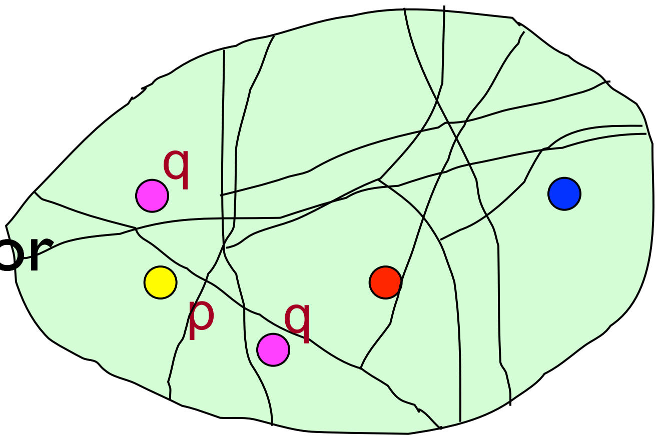
Dimension Reduction: wrap-up

- $F(x) = (g_1 \cdot x, g_2 \cdot x, \ldots g_k \cdot x) / \sqrt{k} = 1/\sqrt{k}\ Gx$
- $||F(x)|| = (1 \pm \epsilon)||x||$ with high probability
- Beyond:
  - Can use ± 1 instead of Gaussians [AMS'96, Ach'01, TZ'04…]
  - Fast JL: can compute faster than in $O(kd)$ time [AC'06, AL'08'11, DKS'10, KN'12…]
  - Other norms, such as ℓ_1 ?
    - 1-stability Cauchy distribution, but heavy tailed!
    - Essentially no: [CS'02, BC'03, LN'04, JN'10…]
    - But will see a useful substitute later!
    - For n points, D approximation: between $n^{\Omega(1/D^2)}$ and $O(n/D)$ [BC03, NR10, ANN10…]

Space Partitions

Locality-Sensitive Hashing

[Indyk-Motwani'98]

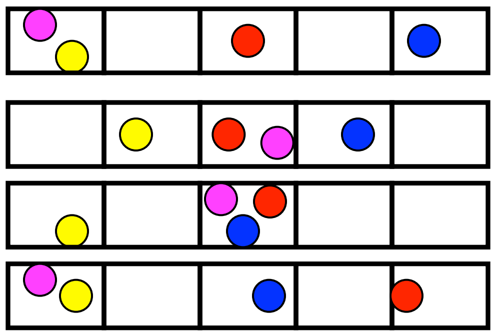


- Random hash function g on $R\uparrow d$ s.t. for any points p,q:
  - Close when $||p-q||\leq r$

$P1=$ $\Pr[g(p)=g(q)]$ is " "not-so-small"

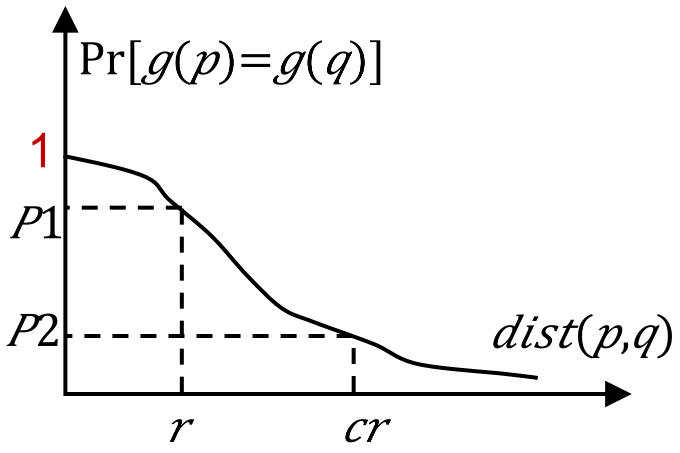
  - Far when $||p-q||>cr$

$P2=$ $\Pr[g(p)=g(q)]$ is "small"



- Use several hash tables : $n\rho$, where

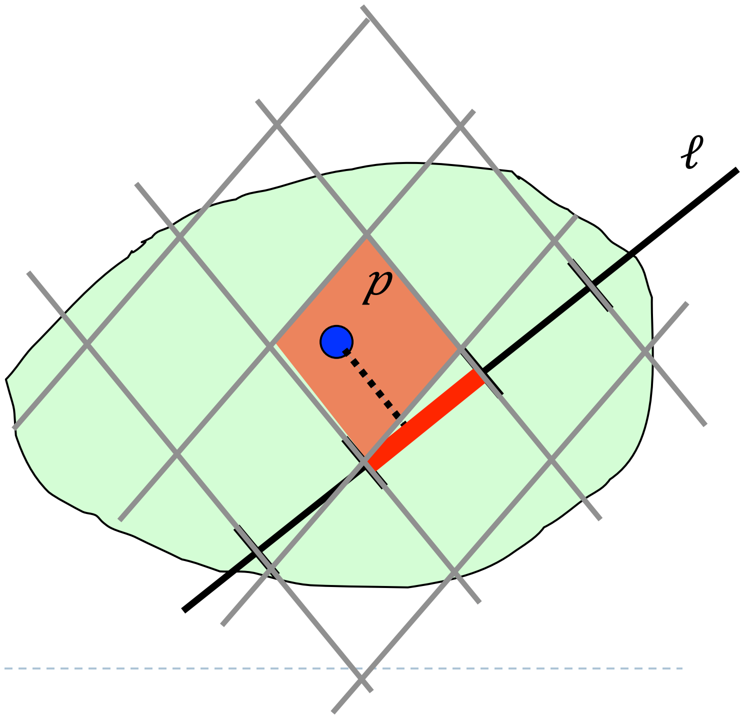
$$P\downarrow 1 = P\downarrow 2 \uparrow \rho$$



NNS for Euclidean space

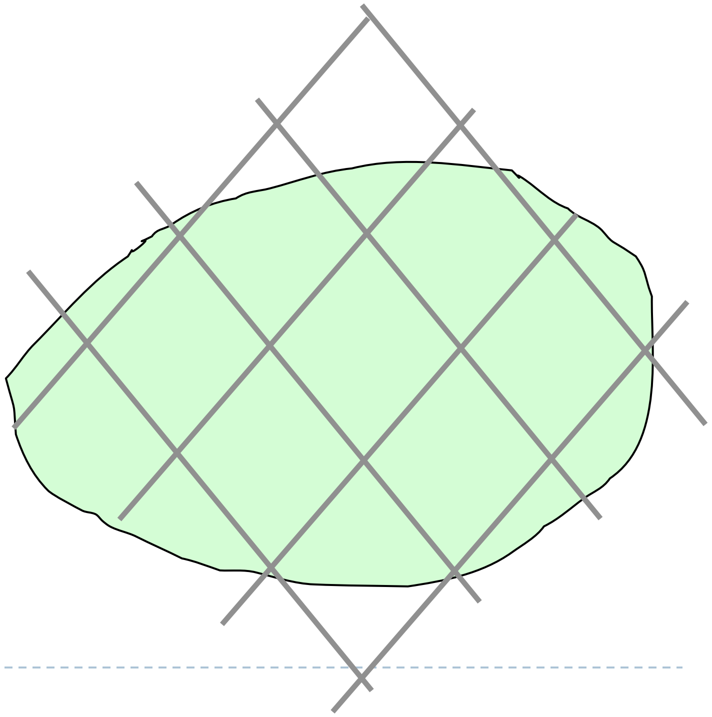
[Datar-Immorlica-Indyk-Mirrokni'04]

- Hash function g is usually 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
  - $h(p)=\lfloor p\cdot\ell/w+b\rfloor$
    - ℓ is a random Gaussian vector
    - b random in $[0,1]$
    - w is a parameter (e.g., 4)
- $\rho=1/c$



Putting it all together

- **Data structure** is just $L = n^{\rho}$ hash tables:
  - Each hash table uses a fresh random function $g(p) = \langle h_1(p), \ldots, h_k(p) \rangle$
  - Hash all dataset points into the table
- **Query**:
  - Check for collisions in each of the hash tables
- Performance:
  - $O(nL) = O(n^{1+1/c})$ space
  - $O(L) = O(n^{1/c})$ query time

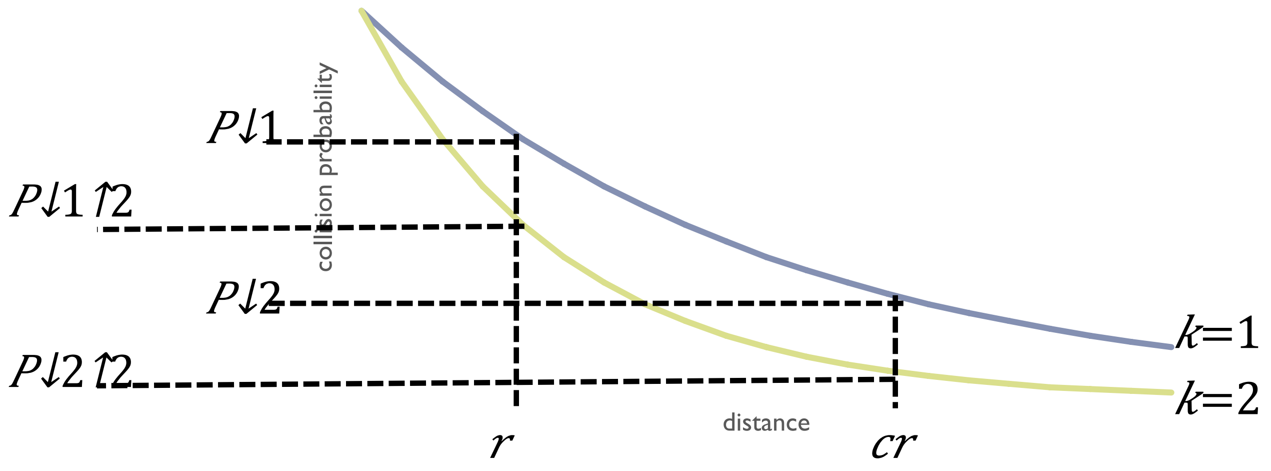


Analysis of LSH Scheme

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

set k s.t.

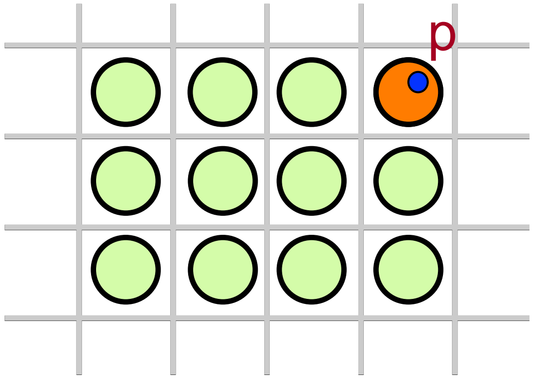
- Pr[collision of *far* pair] $= P_2^k = 1/n$
- Pr[collision of *close* pair] $= P_1^k = (P_2^{\rho})^k = 1/n^{\rho}$
- Hence $L = O(n^{\rho})$ "repetitions" (tables) suffice!



Better LSH ?

[A-Indyk'06]

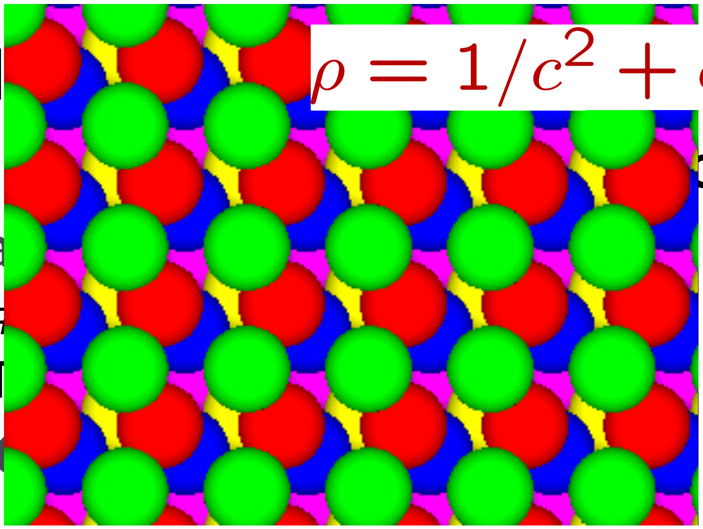
- Regular grid → grid of balls
  - p can hit empty space, so take more such grids until p is in a ball
- Need (too) many grids of balls
  - Start by projecting in dimension t



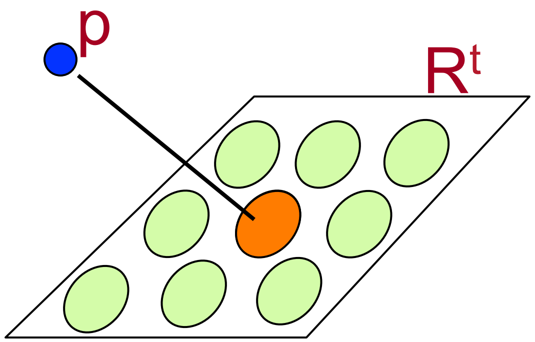
- Anal
- Cho on t?

2D

  - Tra
    - #
    - T
  - Tot



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



Proof idea

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

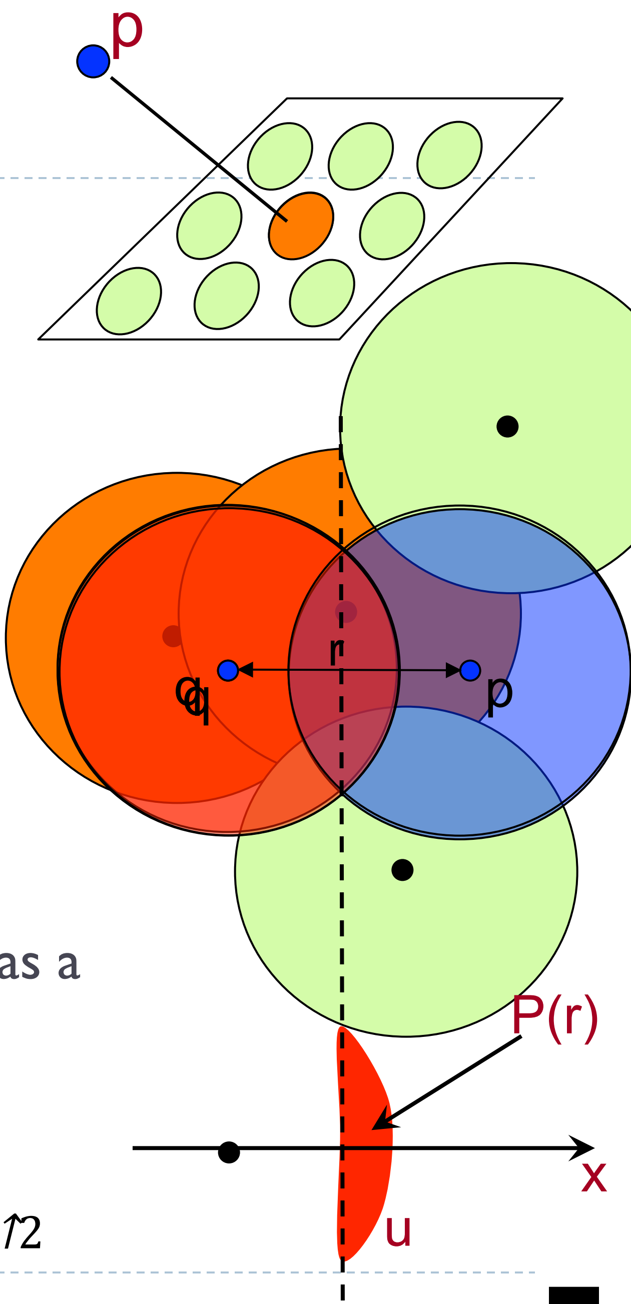
$$P(r) \geq P(cr)^{1/c^2}$$

  - 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

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

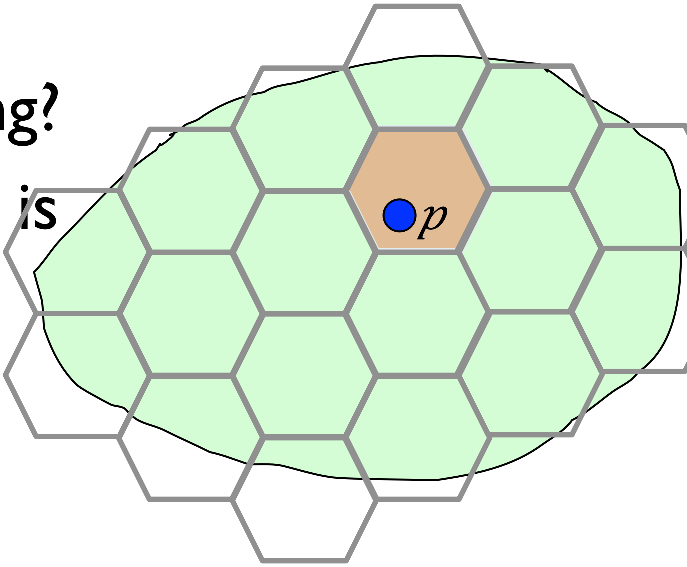
$$P(r)=\exp(-Ar^2)=[\exp(-A(cr)^2)]^{1/c^2}=P(cr)^{1/c^2}$$

$\blacksquare$



Open question:

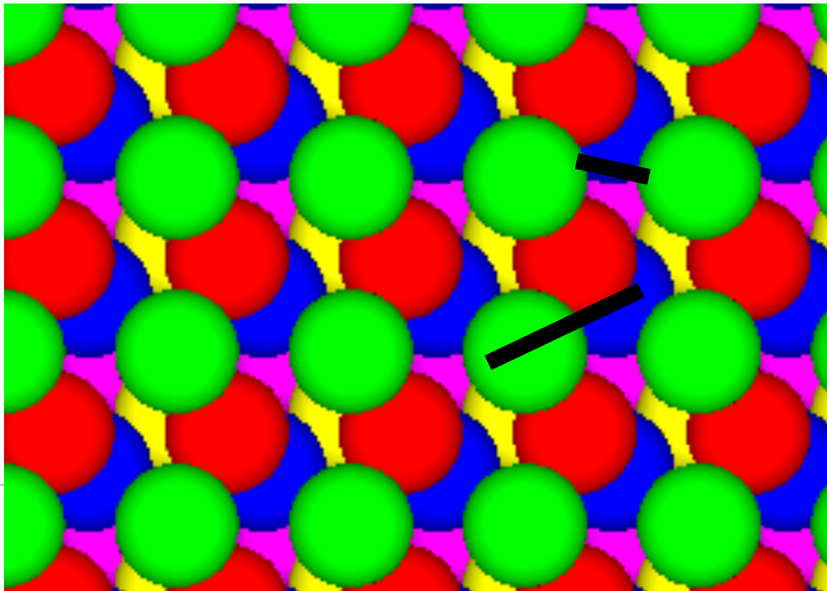
- More practical variant of above hashing?
- Design space partitioning of $\Re^t$ that is
  - efficient: point location in poly(t) time
  - qualitative: regions are "sphere-like"



[Prob. needle of length 1 is not cut]

≥

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



Time-Space Trade-offs

space	query time
low	high
medium	medium
high	low

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

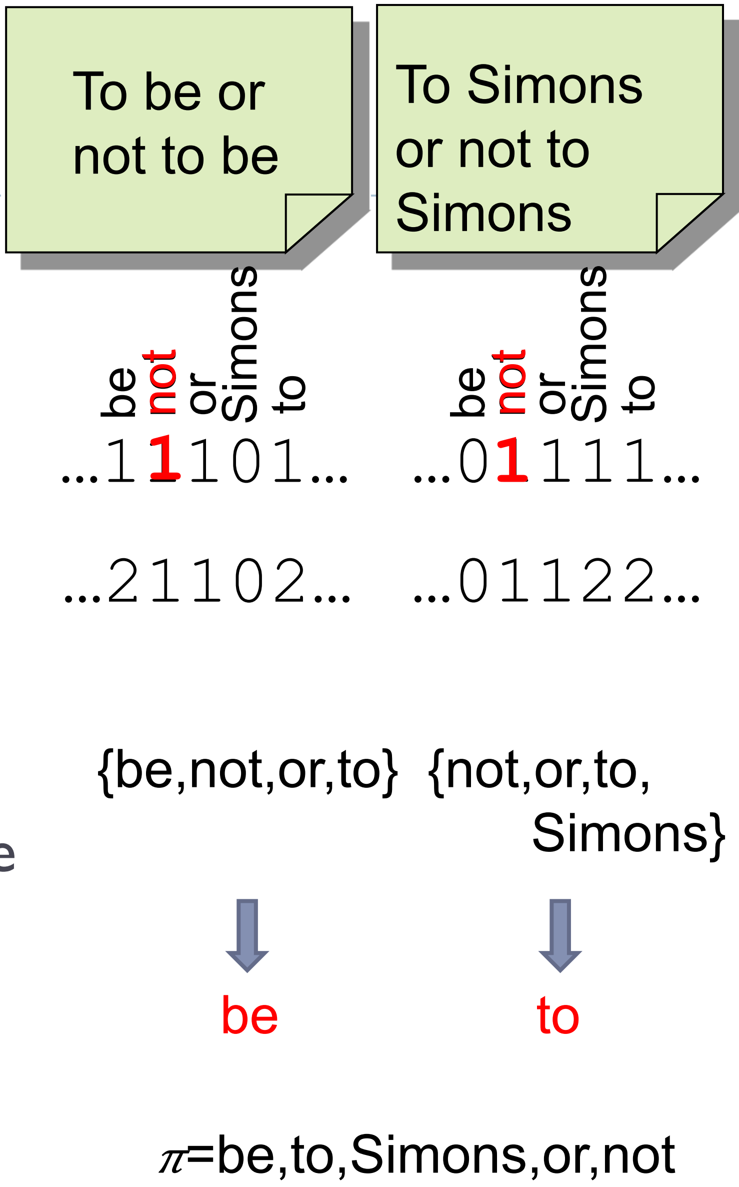
LSH Zoo

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

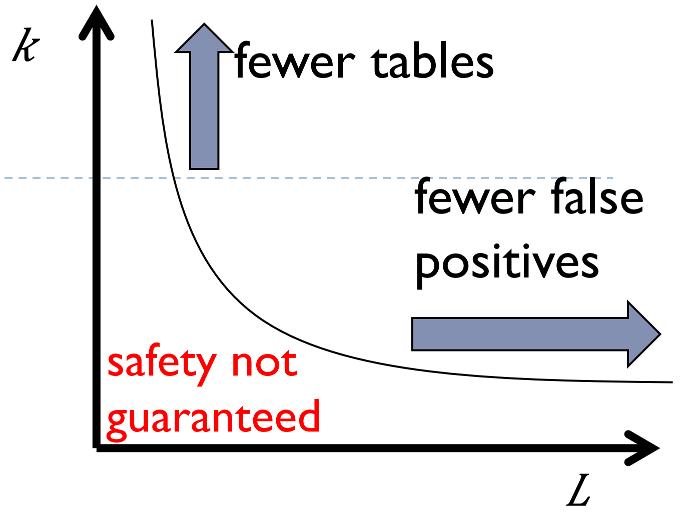
    $h(A)=\min_{a\in A} \pi(a)$

    min-wise hashing [Bro'97,BGMZ'97]

[Cha'02,…]

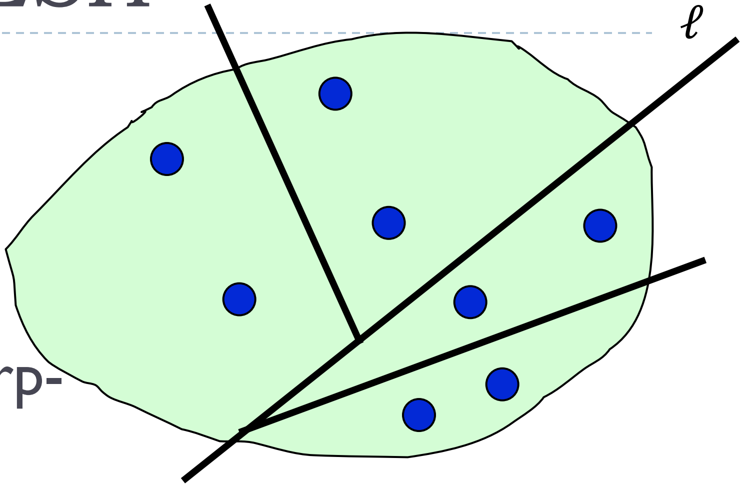


LSH in the wild



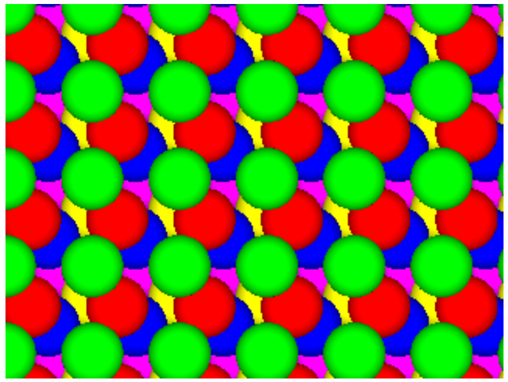
- If want exact NNS, what is c?
  - Can choose any parameters L,k
  - Correct as long as $(1-P_1^k)^L \leq 0.1$
  - Performance:
    - trade-off between # tables and false positives
    - will depend on dataset "quality"
    - Can tune L,k to optimize for given dataset
- Further advantages:
  - Dynamic: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

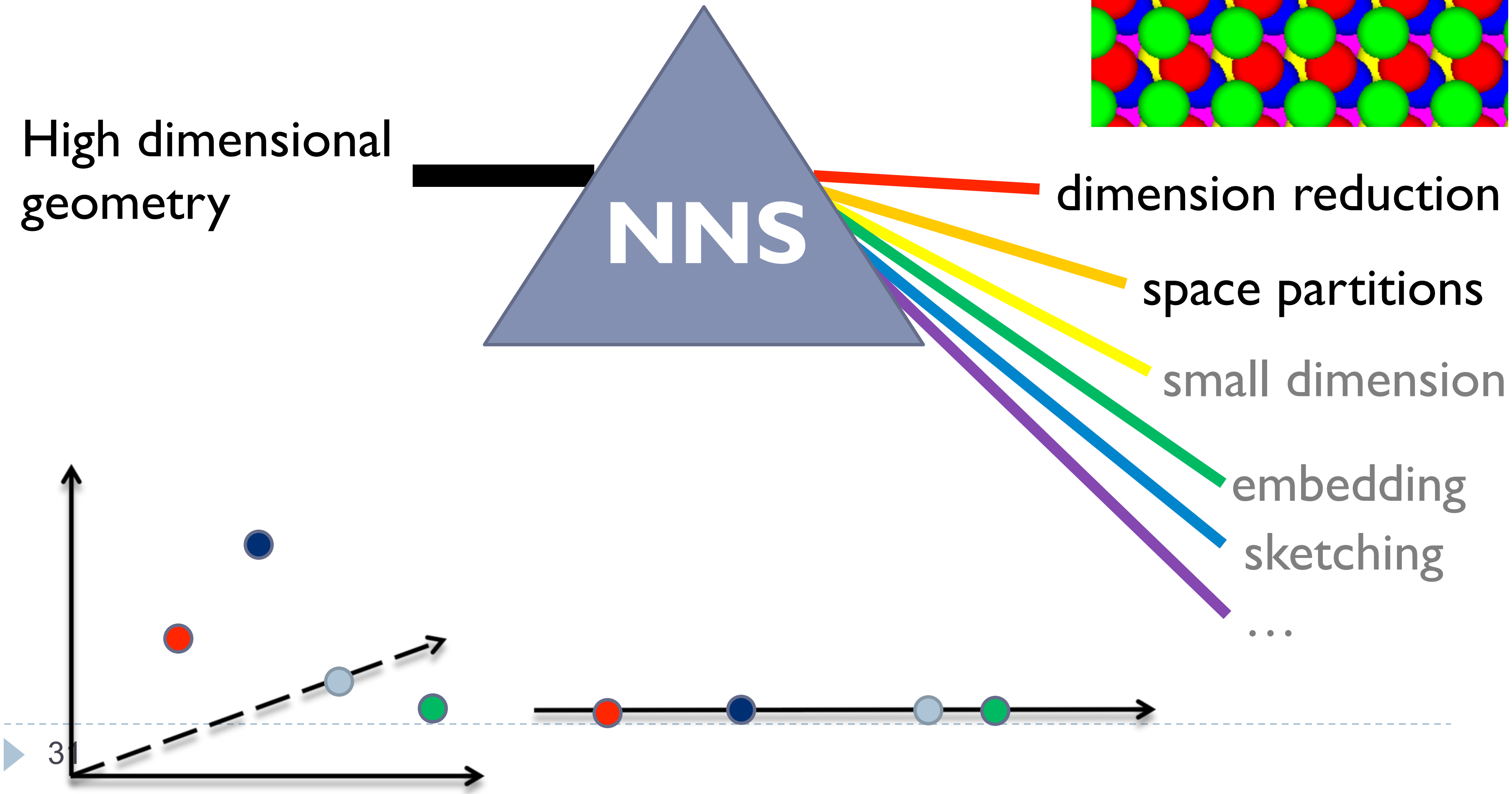
Space partitions beyond LSH



- *Data-dependent* partitions…
- Practice:
  - Trees: kd-trees, quad-trees, ball-trees, rp-trees, PCA-trees, sp-trees…
  - often no guarantees
- Theory:
  - better NNS by *data-dependent* space partitions [A-Indyk-Nguyen-Razenshteyn]
    - $\rho = 7/8/c^2 + O(1/c^3)$ for ℓ_2 cf. $\rho = 1/c^2$ [AI'06, OWZ'10]
    - $\rho = 7/8/c + O(1/c^{3/2})$ for ℓ_1 cf. $\rho = 1/c$ [IM'98, OWZ'10]

Recap for today




High dimensional geometry
NNS
dimension reduction
space partitions
small dimension
embedding
sketching
…